Yapakit
Fortran Editor

Yapakit User's Guide

Michaël Baudin

09/03/2008

Abstract

This report is a User's Guide for Yapakit v1.0. The first part is a small tutorial for Yapakit. The second part is devoted to the software support : the portability of the editor is detailed and the installation process is fully presented. The third part of this document is presenting the main editing features. The final part of this guide is devoted to Fortran features. Many Yapakit features are illustrated by an extensive set of screenshots.

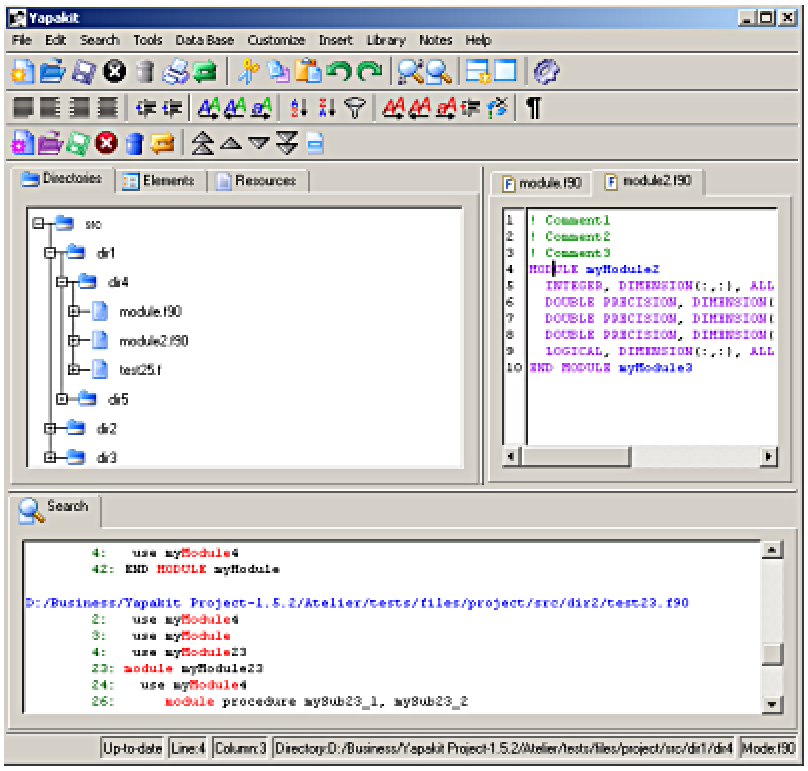
Contents

Introduction

Yapakit is a free, easy to install, easy to use, Fortran source code editor. The goal of this editor is to provide an integrated development environment for Fortran projects. It provides a tool to ease the navigation into the Fortran basic units (function, subroutine, derived- type, etc...) which make a project. The target projects are Fortran 77 and Fortran 90, from small projects (< 10 000 lines of code) to large size projects (> 1 000 000 lines of code).

The current state of this editor is ”experimental” and should be considered as a beta version.

The goal of this chapter is to provide an overview of the current features of Yapakit. In the first section, we describe the main graphical widgets available, the main one being the editing pane. Then we give reasons why to use Yapakit, and reasons why not to use Yapakit. In the 4th section, we describe the main features of Yapakit and in the 5th section we describe the features designed specifically for Fortran.


Yapakit
File Edit Search Tools Data Base Customize Insert Library Notes Help
Directories
Elements
Resources
src
dir1
dir4
module.f90
module2.f90
test25.f
dir5
dir2
dir3
module.f90
module2.f90
Search
Up-to-date
Line:4
Column:3
Directory:D:/Business/Yapakit Project-1.5.2/Atelier/tests/files/project/src/dir1/dir4
Mode:f90

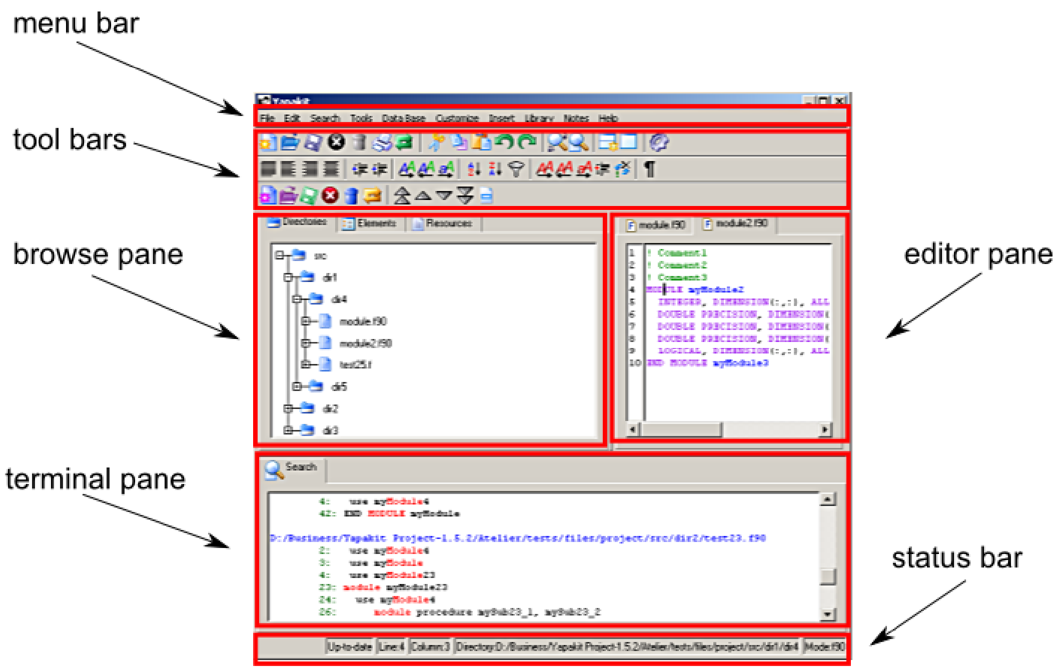
```
! Comment1
! Comment2
! Comment3
MODULE myModule2
  INTEGER, DIMENSION(:,:), ALL
  DOUBLE PRECISION, DIMENSION(
  DOUBLE PRECISION, DIMENSION(
  DOUBLE PRECISION, DIMENSION(
  LOGICAL, DIMENSION(:,:), ALL
END MODULE myModule3
```

```
        4:   use myModule4
        42: END MODULE myModule

D:/Business/Yapakit Project-1.5.2/Atelier/tests/files/project/src/dir2/test23.f90
        2:   use myModule4
        3:   use myModule
        4:   use myModule23
        23: module myModule23
        24:   use myModule4
        26:       module procedure mySub23_1, mySub23_2
```

Overview

In this section, we describe the main graphical widgets with which the user can interact. We first detail each zone of the editor's workspace. The menubar and the 3 toolbars are also detailed.

Editor Workspace


menu bar
tool bars
browse pane
terminal pane
editor pane
status bar
Yapakit
File Edit Search Tools Data Base Customize Insert Library Notes Help
Directories Elements Resources
src
dir1
dir4
module.f90
module2.f90
test25.f
dir5
dir2
dir3
module.f90 module2.f90
Search
Up-to-date Line:4 Column:3 Directory:D:/Business/Yapakit Project-1.5.2/Atelier/tests/files/project/src/dir1/dir4 Mode:f90

```
! Comment1
! Comment2
! Comment3
MODULE myModule2
  INTEGER, DIMENSION(:,:), ALL
  DOUBLE PRECISION, DIMENSION(
  DOUBLE PRECISION, DIMENSION(
  DOUBLE PRECISION, DIMENSION(
  LOGICAL, DIMENSION(:,:), ALL
END MODULE myModule3
```

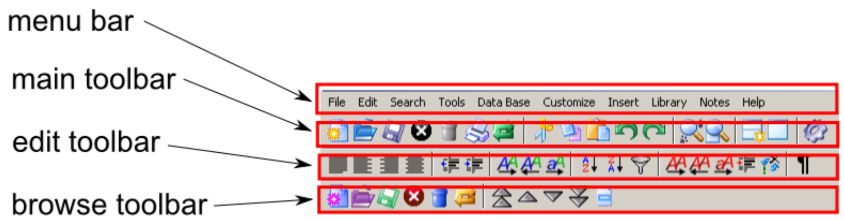
```
        4:    use myModule4
        42: END MODULE myModule

D:/Business/Yapakit Project-1.5.2/Atelier/tests/files/project/src/dir2/test23.f90
        2:    use myModule4
        3:    use myModule
        4:    use myModule23
        23: module myModule23
        24:    use myModule4
        26:       module procedure mySub23_1, mySub23_2
```

Yapakit workspace can be divided into several zones.

- The menu bar let the user access to the main features of the editor.
- The toolbar zone contains up to 3 toolbars.
- The editor pane lets the user edit the files.
- The browse pane contains trees which allow to dynamically explore the content of a Fortran project. It can also contain listbox widgets, which allow for example to display the list of files matching a given file name pattern.
- The terminal pane contains the result of the searches made by the user. It can also contain the messages generated by the commands run in the editor.

Menubar and toolbar

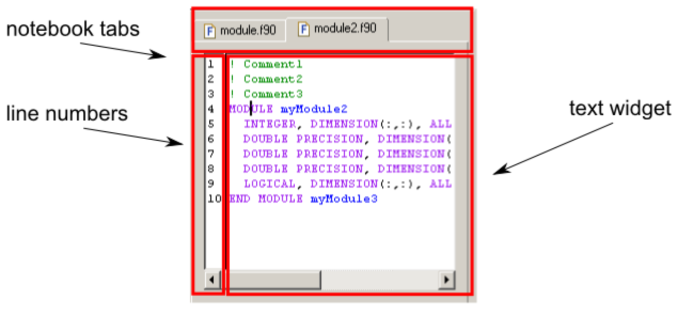
The top of the editor is containing the menubar and the toolbars. Each menu in the menubar can be displayed with a left-click on the mouse. The ”File” menu can also be displayed with the keyboard with ”Alt-f”.


menu bar
main toolbar
edit toolbar
browse toolbar
File Edit Search Tools Data Base Customize Insert Library Notes Help

The toolbars can be displayed or hidden, depending on the choice of the user. By default, all toolbars are displayed. The "Customize > Main toolbar" (resp. "Edit toolbar" and "Browse toolbar") command allows to hide the main (top) toolbar if is is displayed, and allows to display it if it is hidden.

Editing pane

The editing pane contains a Notebook widget which enables to edit several files which can be displayed whenever the user clicks on the corresponding tab. Each tab contain an icon and the file tail of the file under edit. The icon depends on the type of file, identified by its extension, so it is important that the file extensions are consistent with the file content. For example, the icon associated with the fortran files is a blank rectangle containing a blue "F" letter. If the file is modified and unsaved, a red bullet is displayed instead of the icon, to let the user remember that the file has to be saved.



```
1  ! Comment1
2  ! Comment2
3  ! Comment3
4 MODULE myModule2
5   INTEGER, DIMENSION(:,:), ALL
6   DOUBLE PRECISION, DIMENSION(
7   DOUBLE PRECISION, DIMENSION(
8   DOUBLE PRECISION, DIMENSION(
9   LOGICAL, DIMENSION(:,:), ALL
10 END MODULE myModule3
```

The editing zone can be divided into two or more panes, either horizontally or vertically, which authorizes to edit two files at the same time. The "Customize > New pane" command creates a new editing pane : there is no limit in the number of pane which can be displayed, although the graphical space is obviously limited, depending on the screen of the user. The "Customize > Delete pane" command deletes the last pane. In the general customize dialog (Customize > Settings > General), if the checkbox "Pane horizontal" is checked, then each editing pane are inserted horizontally into the editing zone. If the checkbox is unchecked, then each pane is inserted vertically.

Why you should use Yapakit

The following is a list of reasons to use Yapakit specifically instead of another editor or environment.

- It is free, easy to install and easy to use.
- Navigate in a small or large ($> 10^6$ code lines) Fortran project.
- Manages both Fortran 77 and Fortran 90.
- Generates automatically an html documentation of your project.
- Convert your fixed format Fortran 77 into Fortran 90 free source code format.
- Helps the reverse-engineering of an existing Fortran project.
- Generate Fortran 90 dependencies for inclusion in a Makefile : takes module dependencies into account.
- It can be used on Windows, Linux or Sun (Sparc).

Why you should not use Yapakit

The following is a list of reasons *not* to use Yapakit. Although these features are not implemented yet, they are planned in a near future.

- There is *no* direct support for team working. Although Yapakit does not include a system for CVS, nor SVN, we provide a plugin which makes the use of TkCvs/TkSvn easy.
- Automatically compile your Fortran projects is *not* possible yet. Yapakit includes features which allow to generate parts of the Makefile, but it is not currently possible to automatically compile a Fortran project.
- Interactively debugging your Fortran projects is *not* possible yet.
- It *cannot* be used on AIX, HP-UX, Sun Solaris for ix86, Mac OS X. This is a problem currently under work.

Main Features

As a standard text editor, Yapakit allows to

- create, edit, save, rename,
- backup,
- trace history,

of one file. The main window can be splitted in two parts, either horizontally or vertically. The user can navigate throw the files thanks to tabs in a Notebook graphical widget.

Yapakit offers also advanced editing features, in addition of the basic upcase, downcase and capitalize. The editor allows unlimited undo or redo, even undo/redo all, in all files. It allows several sorting methods of the lines : increasing, decreasing, unique or not. It offers upcase/downcase/capitalize of the keywords in the source code.

For Fortran files, Yapakit provides

- comment / uncomment,
- automatic indentation,

features for source code. There are several fill / unfill up to the 72- th column and several justify features (plain, left, right, center). The user can also see invisible characters such as blank spaces or tabulations.

The search and replace dialogs have been made has powerful as possible. The search and replace can be done on one file or several files in one time. Regular expressions are taken into account. The filter dialog allows to remove or keep only lines matching a given pattern. The "Search for files" dialog allows to display the files with several sorting methods : by name, by date, by size or by file tails. Fortran Project Views Yapakit offers four main views, all of which can be exported into picture files for inclusion in reports.

The Directories view is a tree which allows to browse throw the directories and the files of the project. It also allows to navigate into files, with the details of the content of the files : subroutines, functions, modules, etc...

The Elements view is a list box which allows a fast navigation in a project made of a large amount of subroutines, functions or modules. The user only has to set the first letters, Yapakit compute the list that match that letters.

The Resources view is a list box which allows a fast navigation in a project made of a large amount of files. The system is the same that the "Elements" view, but for files.

The Outline view is a tree which allows to browse the content of the current file. The user can then navigate throw subroutines or functions and can set the order of apparition of the items in the list.

Fortran Specific Tools

The focus has been mainly made on Fortran.

Yapakit can generate an html documentation of your fortran project. This page allows a static navigation throw directories, files and elements of the project. Furthermore, the comments are automatically extracted from the source, without any modification of the source itself and thanks to a customizable comment extractor algorithm.

Yapakit can display several figures which allows to get a global view of the project and its main features. It allows to see the general properties of the project, the largest files, the smallest files, the least commented files and the lines distribution by type of lines (code, comment, blank). These figures can be exported into several pictures formats for inclusion in a report.

Yapakit make the management of fortran Makefiles more easy. It includes a dependency algorithm which takes into account modules dependencies. That data can be exported into a Makefile as well as a simple list of the objects files to generate.

Yapakit includes a f77 to f90 free format conversion tool, which can be applied either file by files, or to a collection of files stored in one directory, including the sub-directories.

License

whole or in part, for any form of actual or potential commercial gain or consideration.

Contact yapakit.fortran@gmail.com to discuss any redistribution options not covered by this license agreement. Michaël Baudin reserves all rights not expressly granted to you herein.

This clause is intended to protect the rights of Michaël Baudin's intellectual property and to prevent distribution of products that provide competitive functionality to the Software.

2. Termination. This License Agreement is effective until terminated. Michaël Baudin may terminate this License immediately and without prior notice if you breach any term of this License or for any reason whatsoever. In the event of any termination or expiration, you agree to immediately destroy and/or erase the original and all copies of the Software, any accompanying documentation and License Keys and to discontinue their use and you will not retain or store the Software or any copies thereof, in any form or medium.

3. Proprietary Rights. The Software is licensed, not sold, to you. Michaël Baudin reserves all rights not expressly granted to you. Ownership of the Software and its associated proprietary rights, including but not limited to patent and patent applications, are retained by Michaël Baudin. The Software is protected by the copyright laws of Canada and the United States and by international treaties. Therefore, you must comply with such laws and treaties in your use of the Software. You agree not to remove any of Michaël Baudin's copyright, trademarks and other proprietary notices from the Software.

4. Distribution. Except as may be expressly allowed in Section 1, or as otherwise agreed to in a written agreement signed by both you and Michaël Baudin, you will not distribute the Software, either in whole or in part, in any form or medium.

5. Transfer and Use Restrictions. You may not sell, license, sub-license, lend, lease, rent, share, assign, transmit, telecommunicate, export, distribute or otherwise transfer the Software to others, except as expressly permitted in this License Agreement or in another agreement with Michaël Baudin. In order to use the Software you will be required to obtain a License Key and agree to this License Agreement for the use of the Software. You will not disclose or provide access to your License Key to any other person or entity. You must comply with all applicable Canadian and other export control laws in your use of the Software. Except as may be expressly permitted above, you may not modify, reverse engineer, decompile, decrypt, extract or otherwise disassemble the Software.

6. NO WARRANTY. Michaël Baudin MAKES NO WARRANTIES WHATSOEVER REGARDING THE SOFTWARE AND IN PARTICULAR, DOES NOT WARRANT THAT THE SOFTWARE WILL FUNCTION IN ACCORDANCE WITH THE ACCOMPANYING DOCUMENTATION IN EVERY COMBINATION OF HARDWARE PLATFORM OR SOFTWARE ENVIRONMENT OR CONFIGURATION, OR BE COMPATIBLE WITH EVERY COMPUTER SYSTEM. IF THE SOFTWARE IS DEFECTIVE FOR ANY REASON, YOU WILL ASSUME THE ENTIRE COST OF ALL NECESSARY REPAIRS OR REPLACEMENTS.

7. DISCLAIMER. Michaël Baudin DOES NOT WARRANT THAT THE SOFTWARE IS FREE FROM BUGS, DEFECTS, ERRORS OR OMISSIONS. THE SOFT-

Part I

Tutorials

Chapter 1

Tutorial : Navigating in Sample Project

The goal of this tutorial is to show to the new user the basic process to navigate into a Fortran project. In this tutorial, you will learn :

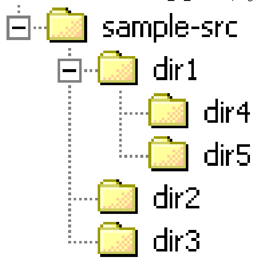
- how to create a database for the project,
- how to navigate in the directories, the files,
- how to navigate in the elements : subroutines, functions, modules,
- how to analyze the whole project with the global properties : number of lines, etc...

1.1 Getting the Fortran sources

The project used in the current tutorial is available as a zip file, containing only Fortran sources. It is available at the following web address :

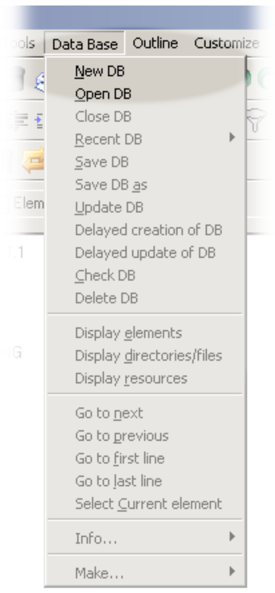
http://perso.orange.fr/yapakit.fortran/tutorialNavigatingInSampleProject-sample-src.zip

These sources have no special computational meaning, they are only here for test purposes. Once unzipped, you should get this directory tree :

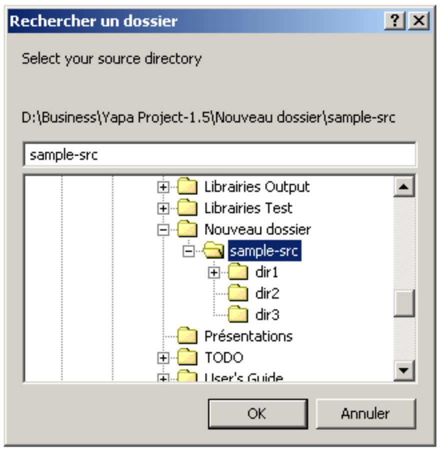


1.2 Creating the Database

To create a new database, go in the ”Database > New DB” menu.



Then select the ”sample-src” directory. This is the directory containing the sources. The directory may contain sub-directories, as in the current example. The editor automatically searches sub-directories for Fortran sources and include the files into the database. Click on ”Ok” to go on to the nex step.

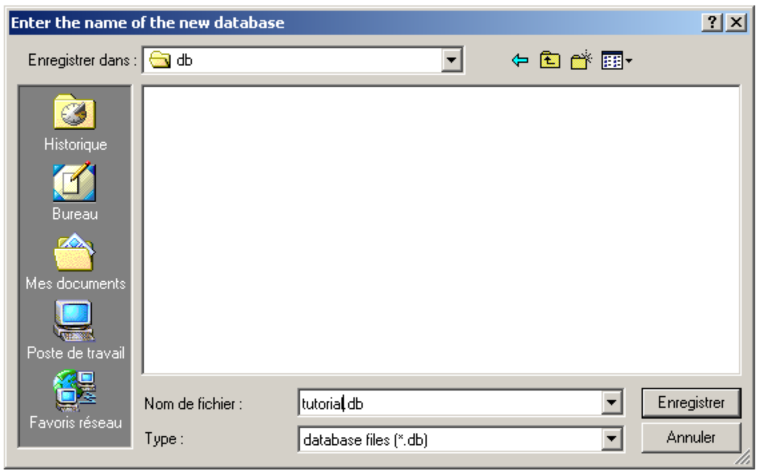

Rechercher un dossier

Select your source directory

D:\Business\Yapa Project-1.5\Nouveau dossier\sample-src

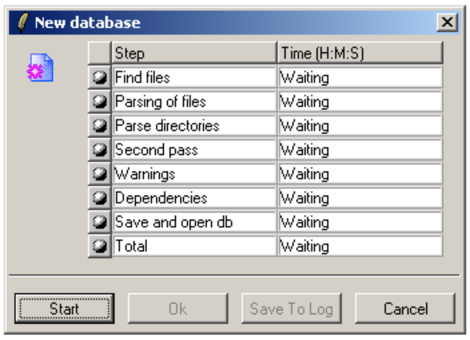
sample-src

The next step is to enter the name of the new database file. Database files are containing all the informations computed during the analysis of the source files. Their extension should be ”.db”. This database file will be automatically loaded after the database is created. It will also be available on disk for the next session.Click on ”Save” to go on to the next step.



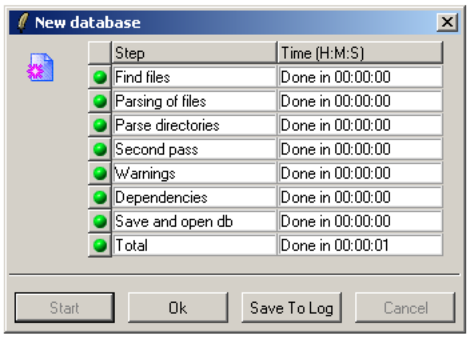
Nom de fichier : tutorial.db
Type : database files (*.db)

The editor then opens the "New database" dialog, which shows the steps which will be processed for the database creation. This dialog allows the user to measure the time needed for the database creation. This time may be above 1 minute for large projects. For the current project, the process needs only a few micro-seconds, depending on your computer. Click on "Start" to go on to the next step.



	Step	Time (H:M:S)
●	Find files	Waiting
●	Parsing of files	Waiting
●	Parse directories	Waiting
●	Second pass	Waiting
●	Warnings	Waiting
●	Dependencies	Waiting
●	Save and open db	Waiting
●	Total	Waiting

Once the database is created, all buttons are green and the right side of the panel indicates the measured time for each step. The content of this dialog may be saved into a log file. For now, it suffices to click on "Ok" to go on to the next step.

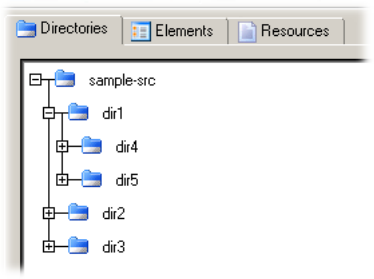


	Step	Time (H:M:S)
🟢	Find files	Done in 00:00:00
🟢	Parsing of files	Done in 00:00:00
🟢	Parse directories	Done in 00:00:00
🟢	Second pass	Done in 00:00:00
🟢	Warnings	Done in 00:00:00
🟢	Dependencies	Done in 00:00:00
🟢	Save and open db	Done in 00:00:00
🟢	Total	Done in 00:00:01

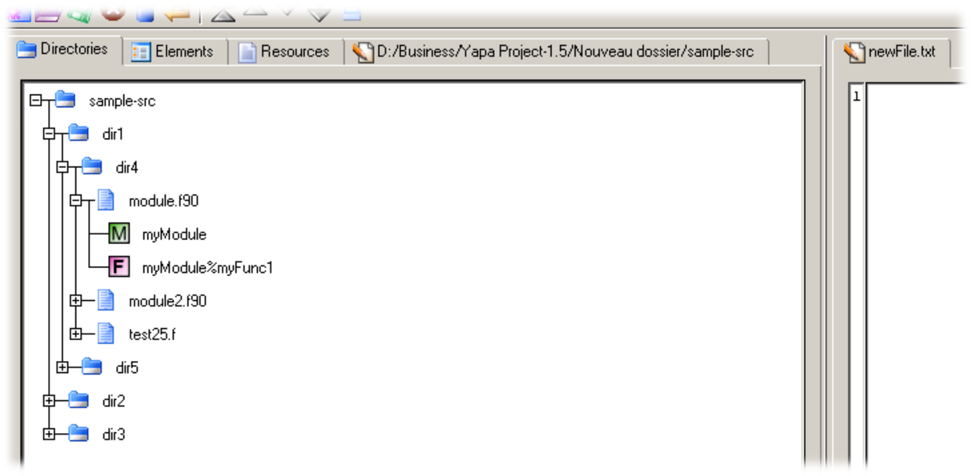
1.3 Navigation in the Project

1.3.1 The Directories View

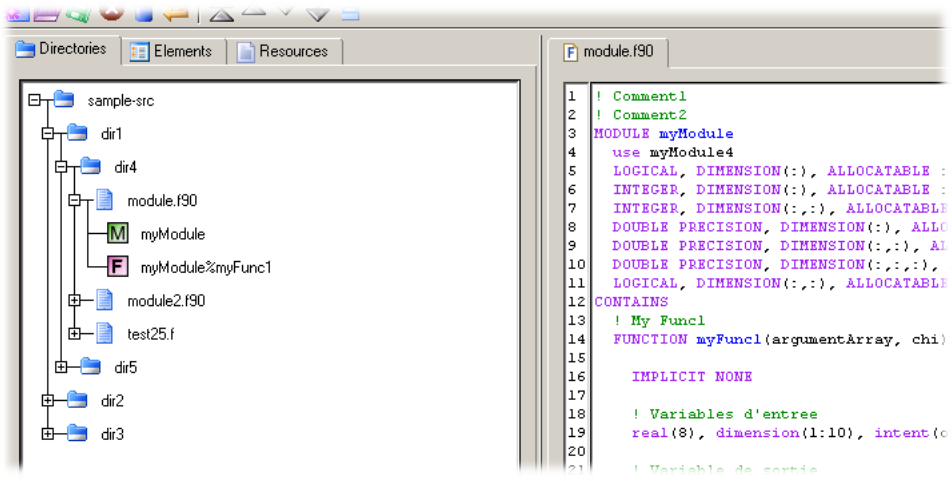
After creating the database, the editor automaticaly creates 3 views : Directories, Elements and Resources. The editor should then look like the following screen capture.



The first thing you might want to do is directly navigating into the project. To do this, left-click on the + sign at the left of the "sample-src" directory. This allows to open the sub-directories. Left-clicking again on "dir1", then "dir4", then "module.f90" allows to see the content of the file, without opening it.

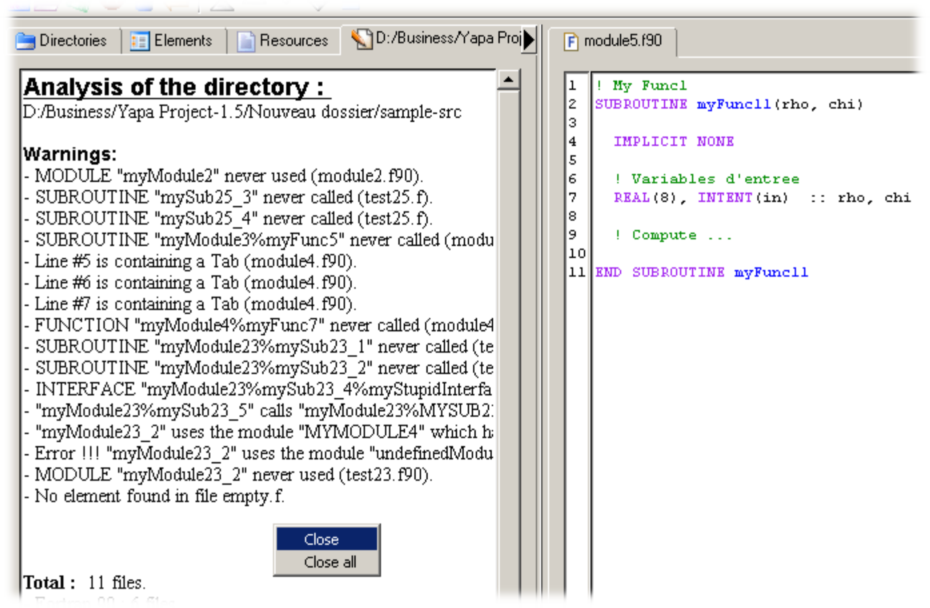


The "M" at the left of "myModule" is because the editor knows that "myModule" is a module. "F" corresponds to a function. If you double-left-click on the "myModule" string of the tree, the editor will automatically open the file, highlight it with the proper Fortran rules, and set the cursor at the line defining the element "myModule". Be sure to "double-left-click" and not to "left-click" to set the cursor. This is because the first left click set the focus on the tree : only the double-left-click allows the defocus the tree to set the focus on the text.



```
! Comment1
! Comment2
MODULE myModule
  use myModule4
  LOGICAL, DIMENSION(:), ALLOCATABLE :
  INTEGER, DIMENSION(:), ALLOCATABLE :
  INTEGER, DIMENSION(:,:), ALLOCATABLE
  DOUBLE PRECISION, DIMENSION(:), ALLO
  DOUBLE PRECISION, DIMENSION(:,:), AL
  DOUBLE PRECISION, DIMENSION(:,:,:),
  LOGICAL, DIMENSION(:,:), ALLOCATABLE
CONTAINS
  ! My Func1
  FUNCTION myFunc1(argumentArray, chi)

    IMPLICIT NONE

    ! Variables d'entree
    real(8), dimension(1:10), intent(o

    ! Variable de sortie
```

In the "Directories" view, if you left-click on the "sample-src" string in the tree, the editor will open a new view, with an analysis of the directory. (Note that left-clicking on the "sample-src" and left-clicking on the "+" or "-" sign in the tree is different.) This view, as other views in fact, can be closed with a right-click. This opens a popup window, which allows to "Close" the current pane or "Close All" panes. Note that these views can be re-opened since they are available in the "Database" menu of the editor. You should not worry about the generated error in the analysis : it is on purpose, to display the possibilites of the analysis.



Analysis of the directory :
D:/Business/Yapa Project-1.5/Nouveau dossier/sample-src

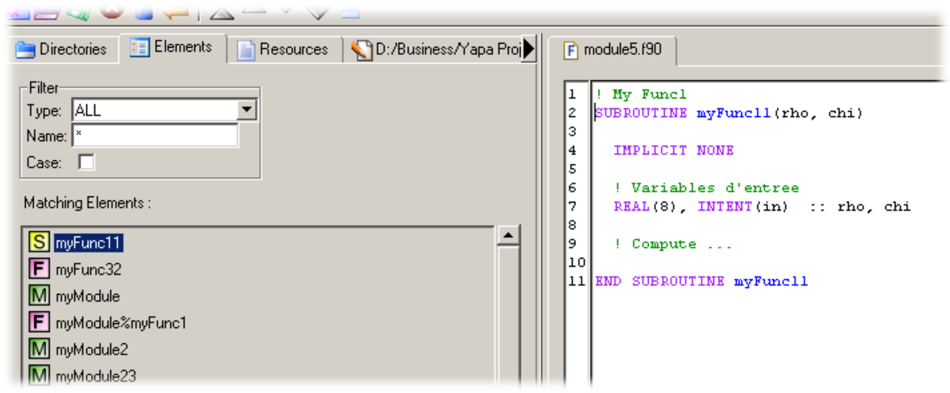
Warnings:
- MODULE "myModule2" never used (module2.f90).
- SUBROUTINE "mySub25_3" never called (test25.f).
- SUBROUTINE "mySub25_4" never called (test25.f).
- SUBROUTINE "myModule3%myFunc5" never called (modu
- Line #5 is containing a Tab (module4.f90).
- Line #6 is containing a Tab (module4.f90).
- Line #7 is containing a Tab (module4.f90).
- FUNCTION "myModule4%myFunc7" never called (module4
- SUBROUTINE "myModule23%mySub23_1" never called (te
- SUBROUTINE "myModule23%mySub23_2" never called (te
- INTERFACE "myModule23%mySub23_4%myStupidInterfa
- "myModule23%mySub23_5" calls "myModule23%MYSUB2
- "myModule23_2" uses the module "MYMODULE4" which h
- Error !!! "myModule23_2" uses the module "undefinedModu
- MODULE "myModule23_2" never used (test23.f90).
- No element found in file empty.f.

Total : 11 files.

```
! My Func1
SUBROUTINE myFunc11(rho, chi)

  IMPLICIT NONE

  ! Variables d'entree
  REAL(8), INTENT(in)  :: rho, chi

  ! Compute ...

END SUBROUTINE myFunc11
```

1.3.2 The Elements View

The Elements view allows to get a full list of the elements in the database. When the view is opened, the "Name" field is left blank. To display all the available elements, type ”*” in the ”Name” field. The editor automatically updates the content of the list everytime it is necessary. The Elements view also allows to filter the elements matching a given type or name pattern. To display only elements starting with letter ”m”, delete ”*” in the ”Name” field and type ”m”. From here, if you double-left-click on the ”myFunc11” element, the editor opens the file ”module5.f90” and set the cursor at the line defining the subroutine ”myFunc11”.



Filter
Type: ALL
Name: *
Case: ☐

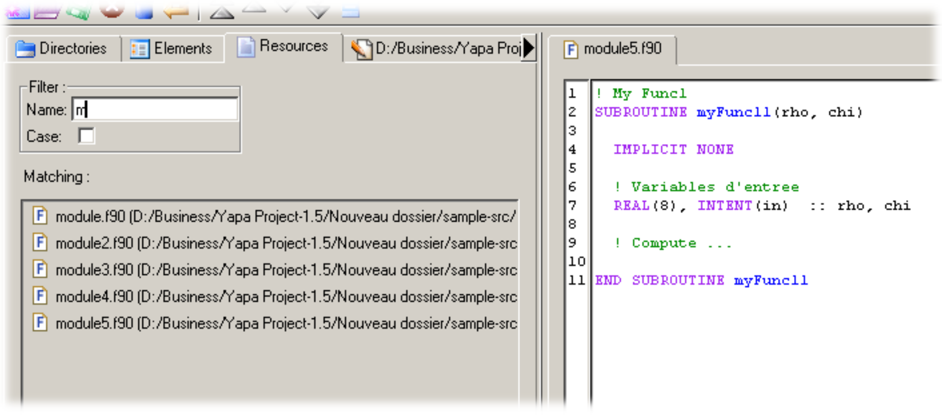
Matching Elements :

S myFunc11
F myFunc32
M myModule
F myModule%myFunc1
M myModule2
M myModule23

```
! My Func1
SUBROUTINE myFunc11(rho, chi)

  IMPLICIT NONE

  ! Variables d'entree
  REAL(8), INTENT(in)  :: rho, chi

  ! Compute ...

END SUBROUTINE myFunc11
```

1.3.3 The Resources View

The Resources view allows to display the list of files of the database, no depend of the directory in which it is located. For example, to display the list of files begining with ”m”, type ”m” in the ”Name” field of the filter, and the editor automatically updates the list of files matching the ”m*” file pattern.



Filter :
Name: m
Case: ☐

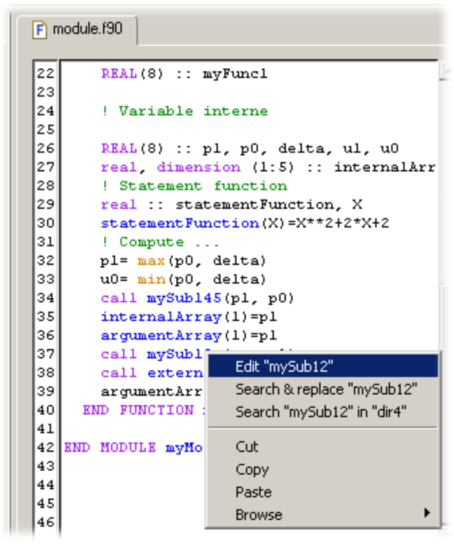
Matching :

F module.f90 (D:/Business/Yapa Project-1.5/Nouveau dossier/sample-src/
F module2.f90 (D:/Business/Yapa Project-1.5/Nouveau dossier/sample-src
F module3.f90 (D:/Business/Yapa Project-1.5/Nouveau dossier/sample-src
F module4.f90 (D:/Business/Yapa Project-1.5/Nouveau dossier/sample-src
F module5.f90 (D:/Business/Yapa Project-1.5/Nouveau dossier/sample-src

```
! My Func1
SUBROUTINE myFunc11(rho, chi)

  IMPLICIT NONE

  ! Variables d'entree
  REAL(8), INTENT(in)  :: rho, chi

  ! Compute ...

END SUBROUTINE myFunc11
```

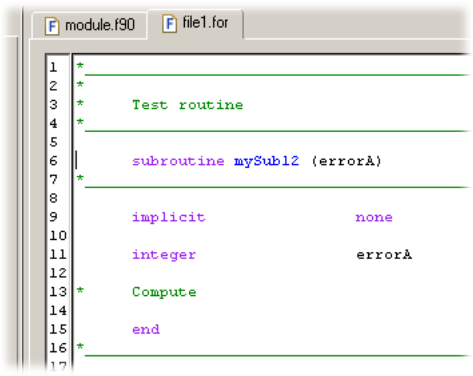
1.3.4 Right-Click Navigation

While editing the ”module.f90” file, you should see that myFunc1 calls the subroutine ”mySub12” at line #37. From there, set the cursor on the ”mySub12” string with a left-click and right-click on your mouse. This opens a popup window which contains the ”Edit mySub12” command, which, if chosen, directly opens the file ”file1.for” and sets the cursor at line #6, at the definition of the subroutine ”mySub12”.



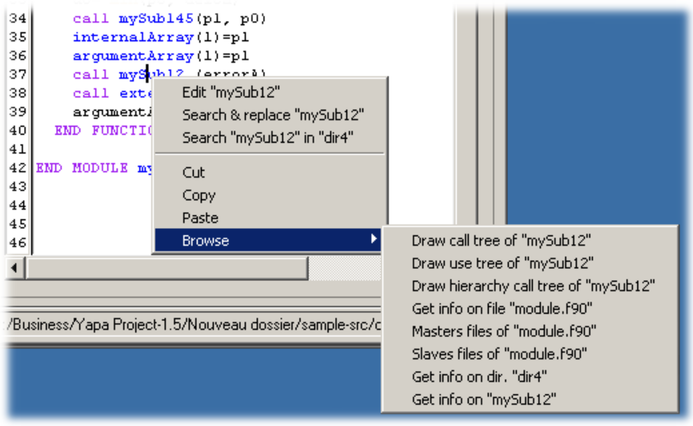
```
    REAL(8) :: myFunc1

    ! Variable interne

    REAL(8) :: p1, p0, delta, u1, u0
    real, dimension (1:5) :: internalArr
    ! Statement function
    real :: statementFunction, X
    statementFunction(X)=X**2+2*X+2
    ! Compute ...
    p1= max(p0, delta)
    u0= min(p0, delta)
    call mySub145(p1, p0)
    internalArray(1)=p1
    argumentArray(1)=p1
    call mySub1
    call extern
    argumentArr
  END FUNCTION :

END MODULE myMo
```

Notice that the source was a Fortran 90 code and that the target is a Fortran 77 code.



```
*
*
*      Test routine
*

       subroutine mySub12 (errorA)
*

       implicit                 none

       integer                  errorA

*      Compute

       end
*
```

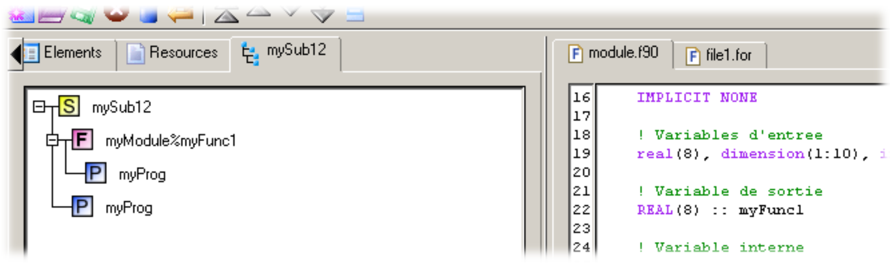
Back to the "module.f90" file, one can again right on the "mySub12" string, but this time opens the "Browse" menu of the popup windows. From there, click on "Draw hierarchy call tree of mySub12" to display a tree allowing to navigate into the reverse call tree of mySub12.



```
    call mySub145(p1, p0)
    internalArray(1)=p1
    argumentArray(1)=p1
    call mySub12 (errorA)
    call exte
    argument
  END FUNCTI

END MODULE my
```

Edit "mySub12"
Search & replace "mySub12"
Search "mySub12" in "dir4"
Cut
Copy
Paste
Browse
Draw call tree of "mySub12"
Draw use tree of "mySub12"
Draw hierarchy call tree of "mySub12"
Get info on file "module.f90"
Masters files of "module.f90"
Slaves files of "module.f90"
Get info on dir. "dir4"
Get info on "mySub12"
/Business/Yapa Project-1.5/Nouveau dossier/sample-src/c

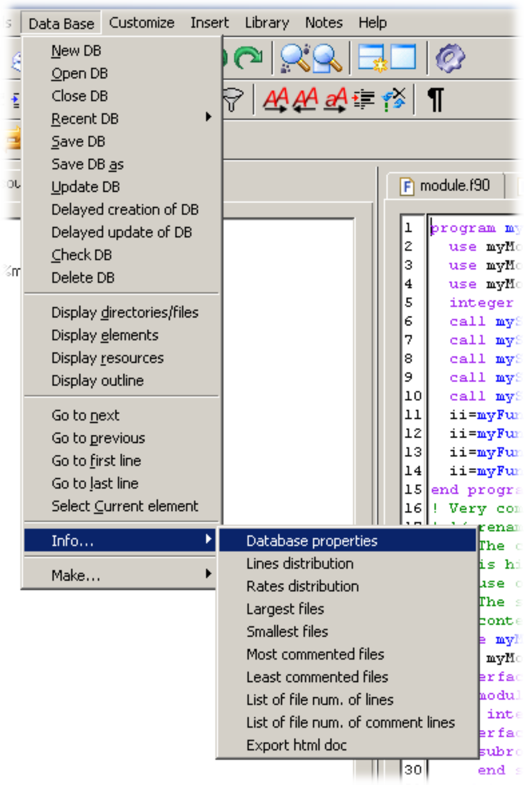
The hierarchy call tree is the reverse of the call tree. Here, mySub12 is called by myFun1 (in the module myModule), which is called by myProg. From there, if you double-left-click on the element myProg, the editor opens the file "test23.f90" and set the cursor at line #1.



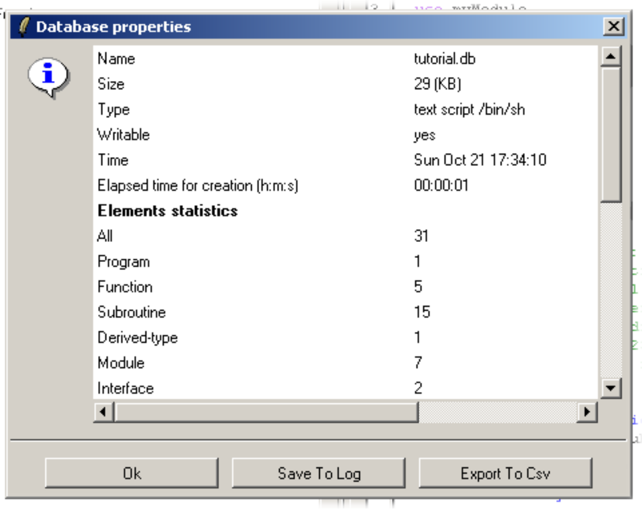
```
    IMPLICIT NONE

    ! Variables d'entree
    real(8), dimension(1:10), i

    ! Variable de sortie
    REAL(8) :: myFunc1

    ! Variable interne
```

1.4 Global Analysis of the project

The best way to get global informations of a new Fortran project is to get an eye on the database properties. Click in the "Database > Info > Database properties" menu.



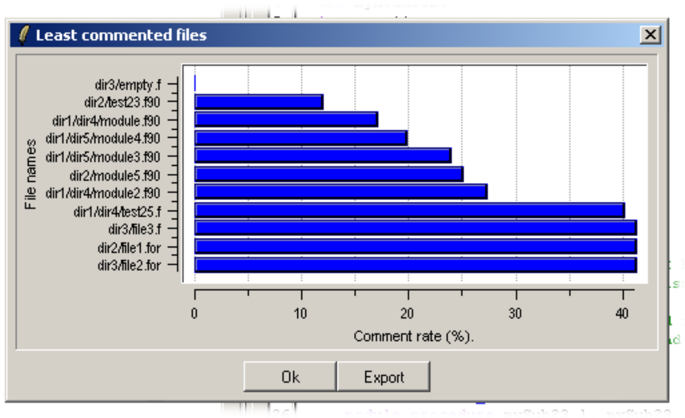
The ”Database properties” dialog allows to get an overview of the project. It displays, for example the total number of programs, the number of files etc... The content of the dialog can be exported, either into a log file or into a csv formatted file (which can be used with a table editor, like Open Office or Excel for example.).



Name	tutorial.db
Size	29 (KB)
Type	text script /bin/sh
Writable	yes
Time	Sun Oct 21 17:34:10
Elapsed time for creation (h:m:s)	00:00:01
Elements statistics	
All	31
Program	1
Function	5
Subroutine	15
Derived-type	1
Module	7
Interface	2

The ”Database > Info Display least commented files” displays a bar graph with the files sorted depending on their comment rate. Obviously, the file ”empty.f” is not commented, but it does not matter. More difficult is the case of ”test23.f90” with only

10% comment rate. The comment rate is a quality indicator, but good quality code requires less comments.



Part II

Support

Chapter 2

Portability

The goal of the Yapakit editor was to be cross-platform because the fortran developers usually handle different systems where their (scientific) softwares are used. The Tcl language was therefore a language of choice because Tcl has been ported on most systems and provides a uniform interface for theses specific platforms. The portability of Yapakit is therefore a ”by design” feature.

2.1 Supported systems

The goal of this section is to detail the plaforms where Yapakit is available and where it is not available.

Yapakit is available on the following platforms :

- win32-ix86
- linux-ix86
- solaris-sparc
- solaris-ix86
- hpux-parisc
- aix-rs6000
- macosx

On the base of the Active State Tcl/Tk 8.4 system, Yapakit has been tested under the following operating systems :

- Windows (Windows 98, Windows 2000, Windows ME, Windows XP),
- Linux (Mandrake, Linux Red Hat),
- Sun Solaris (Sparc).

Remark 2.1.1 *Previous releases of Yapakit included the BLT package, which was used in a compiled form. Because of this, Yapakit was not available on the platforms where BLT was not available. Yapakit is now based on the "Plotchart" package, part of the Tklib project, a work of Arjen Markus.*

2.2 Yapakit StarPack

Yapakit is released as a Tcl Starpack and the goal of this section is to describe how this starpack is made.

The Yapakit editor is available as a system-specific executable and a set of additionnal directories. The Yapakit executable contains a Tcl/Tk interpreter, the Yapakit packages and a set of additionnal packages, all of which are stored in a virtual file system. Therefore, the user does not have to get an existing Tcl/Tk system installed on the target system to run Yapakit.

This feature is known as a "Starpack" in the Tcl community. The executable was created with the TclApp software, a part of the Tcl Dev Kit of Active State.

Chapter 3

Known bugs

3.1 Unit tests

This software has been developped using a methodology based on testing. The 500 unit tests are run on each release. More than 1500 source code and text files are under test in the test database. This is no a guaranty for a software without bugs, but is a guaranty that every detected and fixed bug will be consistently fixed in next releases. Such a software would not be possible without a unit test database. The Tcl package ”tcltest” is the base of the unit test framework.

3.2 Known bugs

Although an effort has been made in order to produce a software as free of bug as possible, several detected and undetected bugs remain in Yapakit.

- In fortran files, array are colorised as functions/subroutines, which is wrong. This is because it is not easy to distinguish them : the analysis requires a two-passes algorithm, which is not done in the highlighting process.

- When searching for a string printed on several lines, the tk text search command fails. It is a limitation of the tk widget.

- The parsing of fortran files does not take into account the preprocessor commands. This can lead to a wrong analysis of the source code. For example, some warnings can be displayed (for example, unused subroutine), which are not correct when the user takes the preprocessor instructions into account. This bug could only be corrected if a preprocessor command was used before the parsing of any file. Only the user can give this command to Yapakit since it includes the list of preprocessor macros and a preprocessor executable (like fpp, fortran preprocessor, for example).

3.3 Known missing features

This section details a list of features which are not available in Yapakit yet, but are planned in the future.

- When one file is opened in the editor and then modified on the disk, Yapakit does not inform the user of the modification. It should suggests the user to reload the file, if needed.
- There is no possibility to make a rectangular selection with the mouse. Although it is possible to select particular columns for all lines of one file, there is no possibility to select specific columns for a given range of lines.

Of course, the list of missing features is not complete and users are encouraged to contact the author to ask for missing features that they think would be worth to include.

Chapter 4

Installation

This chapter is devoted to installing Yapakit on a computer. This section contains a quick installation guide, with a detailed list of the supported systems and the environment configuration needed to run Yapakit. The environment configuration is presented in the next section, along with the basic configuration of Yapakit itself. The last two sections deals with Yapakit source code protection and time limitation.

4.1 Quick guide

Yapakit is currently available on the following operating systems :

- win32-ix86
- linux-ix86
- solaris-sparc
- solaris-ix86
- hpux-parisc
- aix-rs6000
- macosx

These are the steps to go through to install and run Yapakit :

- Identify the platform where you want to install and run Yapakit.
- Send an e-mail to yapakit.fortran@gmail.com with *"Yapakit request for platform xxxx"* as the subject. You will get back your license key as a reply within a few hours.
- Download and unzip the zip file containing Yapakit for your platform.

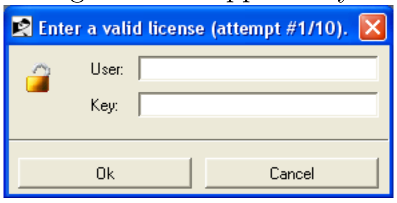
- On windows, double click on ”yapakit.exe” and enter the license key in the dialog.
- On Linux/Unix platforms, make the file ”yapakit-your-platform” executable (with ”chmod u+x yapakit-your-platform”). Then run Yapakit from command line with ”yapakit-your-platform” and enter the license key in the dialog.
- Enjoy !

There is no limit in the number of requests users can do. If you need Yapakit both on Windows and Linux (or any other platform), let me know in the email so that I can send back to you the corresponding keys.

The following is a sample licence key report, the one that you should get by email. It contains the name of the release you are requiring, the target platform, the serial key and the expiration date (see below for details for that particular feature).

```
License key report
Release directory : yapakit-release-20XX.XX.XX.XX.XX.XX-win32-ix86
Platform : win32-ix86
Serial key : XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
Expiration date : XX XX 20XX
```

When running Yapakit for the first time, the following dialog should appear. The ”User” field is optional. The ”Key:” field is mandatory : copy the ”XXXXX-XXXXX-XXXXX-XXXXX-XXXXX” string from the email to the dialog and click OK. The key will be stored into the user’s preferences data file when the session is closed, that is when you quit Yapakit. That way, when you run Yapakit the second time, the key is read and the dialog does not appear anymore.



4.2 Yapakit Environment and Configuration

The installation of a Yapakit on a target platform is meant to be fast and simple. The user simply unzip a platform-specific zip file : installation is done ! The uninstallation of Yapakit is : delete the Yapakit directory ! It cannot be simpler. There is no need for an existing Tcl/Tk system since Yapakit contains its own internal Tcl/Tk interpreter.

The current version of Yapakit is a zip file. You can use for example http://www.7-zip.org to unzip the release. Inside, you will find a directory ”yapakit.release.2007.xx.xx”, containing the full release. The main executable is the file ”yapakit-release-20XX.XX.XX.XX.XX.XX-platform” or ”yapakit-release-20XX.XX.XX.XX.XX.XX-win32-ix86.exe” on Windows systems.

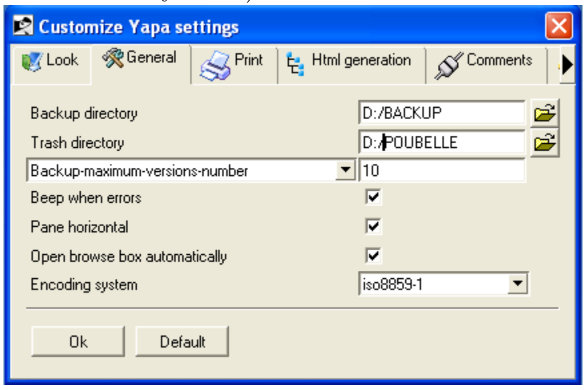
- Under Windows platforms, the user only has to double click on ”yapakit-release-20XX.XX.XX.XX.XX.XX-win32-ix86.exe” to run the software. An option is to create a link to the executable and to put it onto the Windows desktop, but this is not stricly necessary.

- Under Linux / Unix platforms, the user may have to change permissions so that the yapakit file can be executable. This can be done with, for example, the command

  ```
  chmod u+x yapakit-release-20XX.XX.XX.XX.XX.XX-linux-ix86
  ```

  on the Linux system. Simply type ”yapakit-release-20XX.XX.XX.XX.XX.XX-linux-ix86” in a terminal to run the software. An option is to add the path to the directory containing yapakit to the ”PATH” environment variable, but this is not strictly necessary.

The documentation is stored in the ”doc” directory, under the root of the release directory. The ”doc” directory contains the same documentation as the online documentation. During the first session, the user should configure the backup and trash system. The backup system allows to recover older versions of the edited files. The trash system allows to recover the files that were put into the basket case. Go into the Customize menu, select Settings, and go on to the General tab. Set your backup and trash directory, which must be existing directories (the editor does not create the directories if it does not allready exist).



Backup directory: D:/BACKUP
Trash directory: D:/POUBELLE
Backup-maximum-versions-number: 10
Beep when errors: ✓
Pane horizontal: ✓
Open browse box automatically: ✓
Encoding system: iso8859-1

4.3 Uninstall Yapakit

Users who wish to uninstall Yapakit should first remove the directory containing Yapakit, then the ”.yapakit” directory created by Yapakit to store preferences of the user.

- Under Windows platforms, the preferences directory of the user is typically the directory :

  ```
  C:/Documents and Settings/UserName/.yapakit
  ```

- Under Linux platforms, the preferences directory of the user is typically the directory :

```
/home/username/.yapakit
```

An option is to delete also the ”backup” and ”trash” directories, where Yapakit may have created backup files or moved trashed files. This may be particularily useful especially for the ”backup” directory because it may contains a lot of files, each one being a backup of a particular state of the files you edited with Yapakit.

4.4 Protection of Yapakit

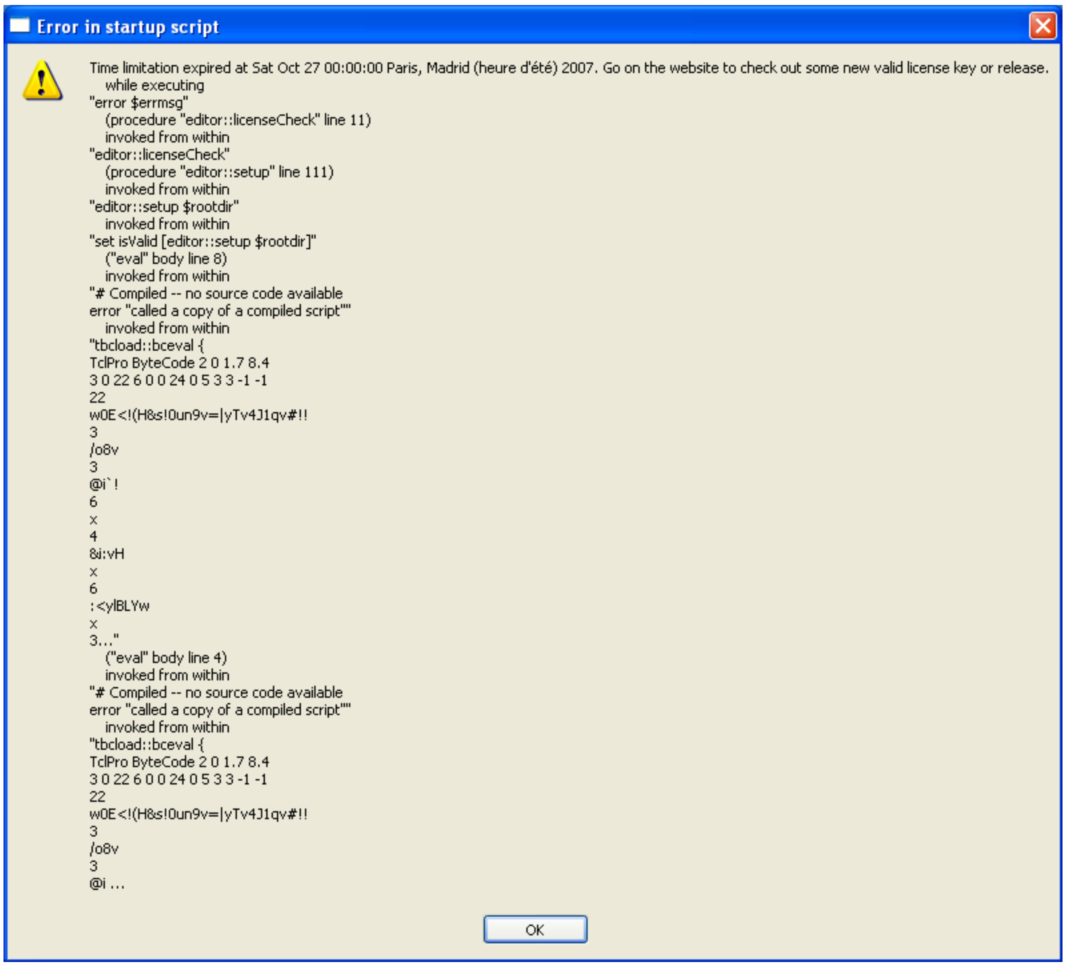
The source code of the software is protected by several techniques which should make at least difficult the task of reverse-engineering it (even with the uncrypted source, it is not allways obvious to reverse-engineer my own source code anyway !). In particular, it uses a Tcl Compiler which produces a bytecode directly usable by the Tcl interpreter.

4.5 Time limitation

Yapakit is in a beta state and many releases are to come, with bug fixes and new features. The current rule is that a release cannot work more than 180 days after it was created.

The time limit forces the users to go back to the Yapakit website to see if some new release is available. Because Yapakit is in beta state, more features and bug fixes are to come in the next months.

If you try to use the editor beyond the time limitation, you will get a message like the following.

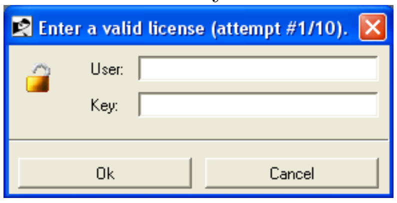


```
Time limitation expired at Sat Oct 27 00:00:00 Paris, Madrid (heure d'été) 2007. Go on the website to check out some new valid license key or release.
    while executing
"error $errmsg"
    (procedure "editor::licenseCheck" line 11)
    invoked from within
"editor::licenseCheck"
    (procedure "editor::setup" line 111)
    invoked from within
"editor::setup $rootdir"
    invoked from within
"set isValid [editor::setup $rootdir]"
    ("eval" body line 8)
    invoked from within
"# Compiled -- no source code available
error "called a copy of a compiled script""
    invoked from within
"tbcload::bceval {
TclPro ByteCode 2 0 1.7 8.4
3 0 22 6 0 0 24 0 5 3 3 -1 -1
22
w0E<!(H&s!0un9v=|yTv4J1qv#!!
3
/o8v
3
@i`!
6
x
4
8i:vH
x
6
:<ylBLYw
x
3..."
    ("eval" body line 4)
    invoked from within
"# Compiled -- no source code available
error "called a copy of a compiled script""
    invoked from within
"tbcload::bceval {
TclPro ByteCode 2 0 1.7 8.4
3 0 22 6 0 0 24 0 5 3 3 -1 -1
22
w0E<!(H&s!0un9v=|yTv4J1qv#!!
3
/o8v
3
@i ...
```

In that case, simply go back to the web site and download an up-to-date release.

4.6 Licence key

The following dialog appears at the first startup of Yapakit, which asks for a new user name and license key.



The field ”User” is meant to be the name of the user, but is not mandatory. The field ”Key” is the licence key, provided by the author of Yapakit by email. This software cannot start without a valid license key. To get a valid license key, simply send an email to yapakit.fortran@gmail.com with ”Yapakit request” as the subject.

The goal of this feature is for the author of Yapakit (me !) to get informations about who really uses the software (you !), not just who downloads it from the web site. Since Yapakit is in beta state, feedback from users will allways be welcomed.

Part III

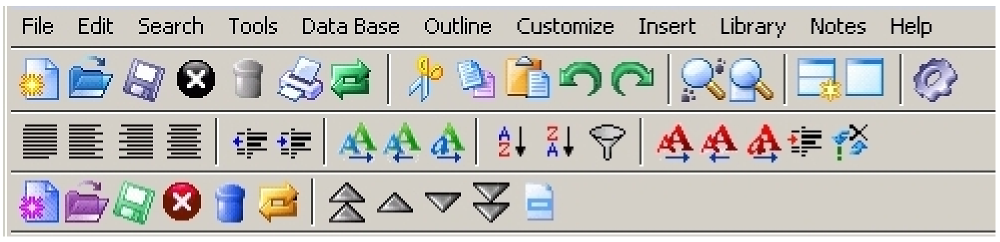
Editor Features

Chapter 5

Overview

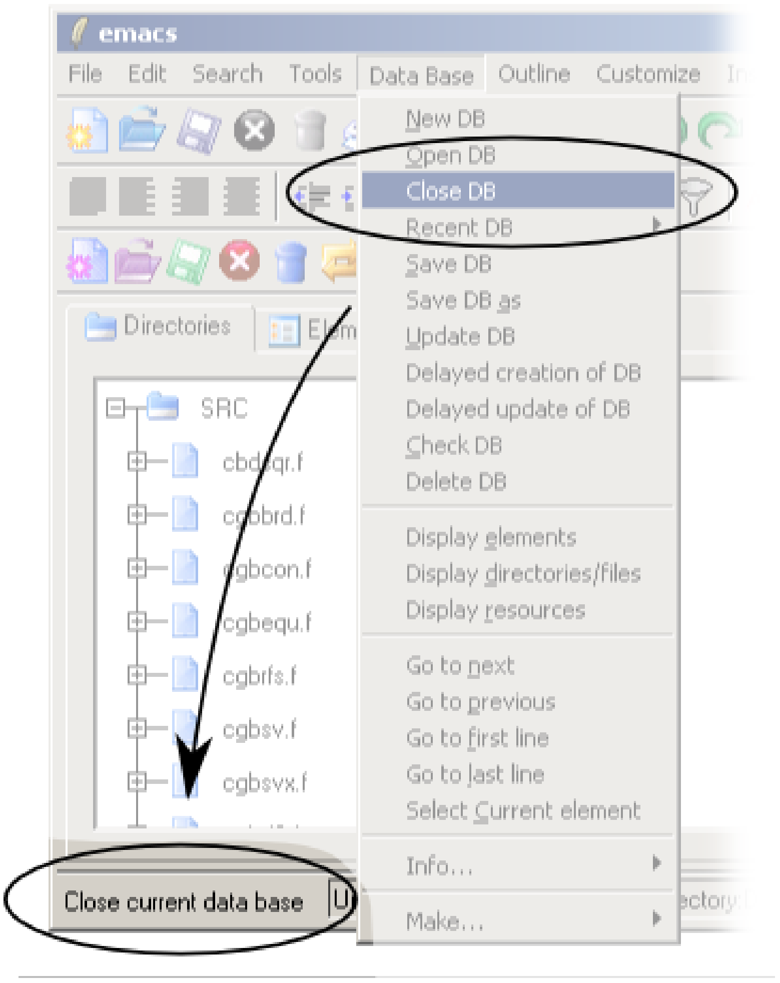
Yapakit is a Fortran editor. The goal of this editor is to provide an integrated developement environnement for Fortran.

5.1 Menu bar and tool bar

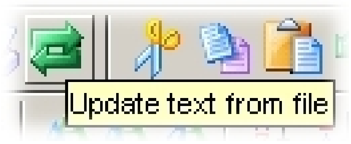


The menubar contains the classical menus "File", "Edit", "Search", "Tools", "Data Base", "Elements", "Insert", "Customize" and "Help". The "File" menu is dedicated to file managing: new file, open file, close file, delete file, rename file, file properties and printing. The "Edit" menu is dedicated to file editing: cut, copy and paste, undo and redo, and text modifications (upcase, downcase and capitalisation indentation etc...). The "Search" menu is helping the user to find and/or replacing words or sentences in the current file or in multiple files. User defined menus can be added thanks to the plugin mechanism.

5.1.1 Toolbars



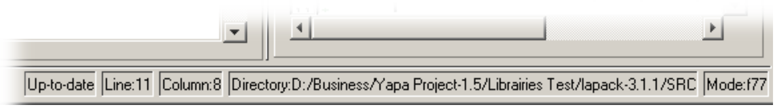
The toolbar is divided into three bars : the main toolbar (new file, open, save, close, ...), the editing toolbar (adjust text, indent, upcase, etc...) and the browse toolbar (new database, open, close, save, delete, update, etc...) All items of the menus are associated with a help text, displayed in the status bar.



All icons displayed in the toolbars are associated with a dynamic balloon help.

5.2 Window managing

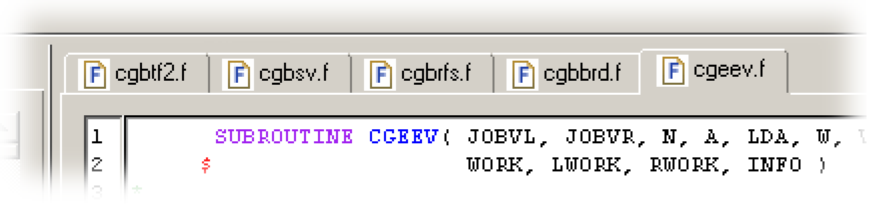
5.2.1 Status bar



```
Up-to-date | Line:11 | Column:8 | Directory:D:/Business/Yapa Project-1.5/Librairies Test/lapack-3.1.1/SRC | Mode:f77
```

The status bar is displayed at the very bottom of the window. The left part of the status bar is devoted to help for menu items. The right part of the status bar is displaying the line and column numbers of the cursor, the directory name for the current file, the mode associated with the current file.

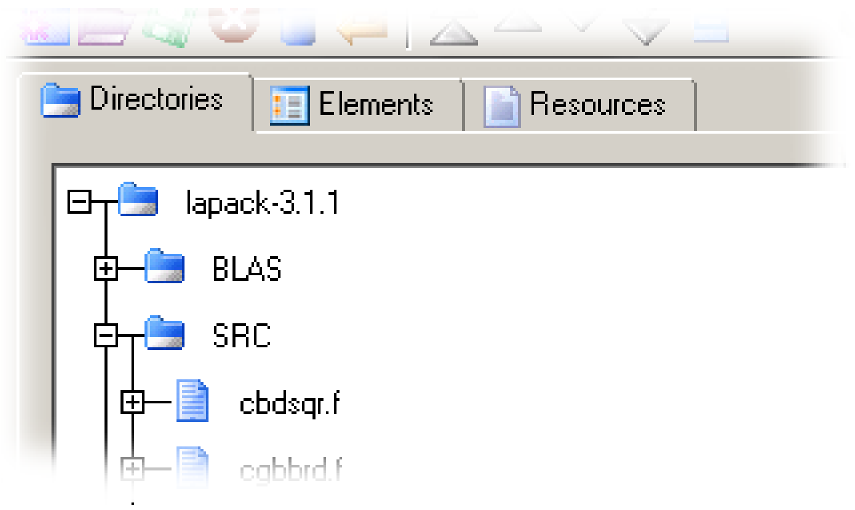
5.2.2 Multiple files



```
      SUBROUTINE CGEEV( JOBVL, JOBVR, N, A, LDA, W,
     $                  WORK, LWORK, RWORK, INFO )
```

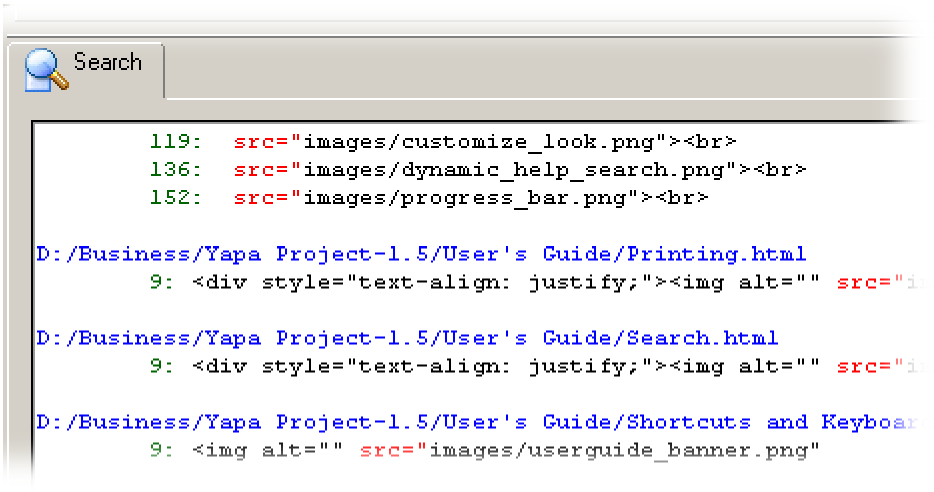
The main part is dedicated to the text editing. Multiple files can be opened simultaneously, and are accessible with tabs in a notebook. The text editing region may be splitted into two or more parts. Panes are added and deleted when desired. This allow to see differents parts of one file at the same time.

5.2.3 Browse pane



At the left of the window, a browse pane is dedicated to the analyse of the content of a database. The user can browse into directories, files and elements of the database.

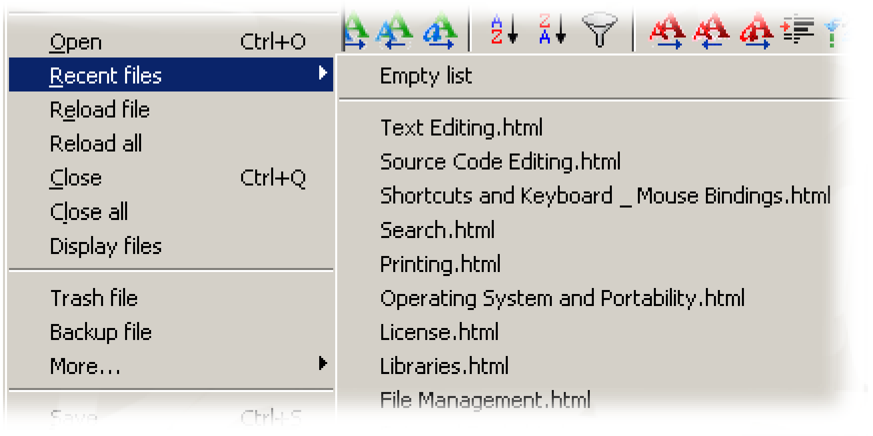
5.2.4 Search terminal



```
        119:  src="images/customize_look.png"><br>
        136:  src="images/dynamic_help_search.png"><br>
        152:  src="images/progress_bar.png"><br>

D:/Business/Yapa Project-1.5/User's Guide/Printing.html
        9: <div style="text-align: justify;"><img alt="" src="i

D:/Business/Yapa Project-1.5/User's Guide/Search.html
        9: <div style="text-align: justify;"><img alt="" src="i

D:/Business/Yapa Project-1.5/User's Guide/Shortcuts and Keyboar
        9: <img alt="" src="images/userguide_banner.png"
```

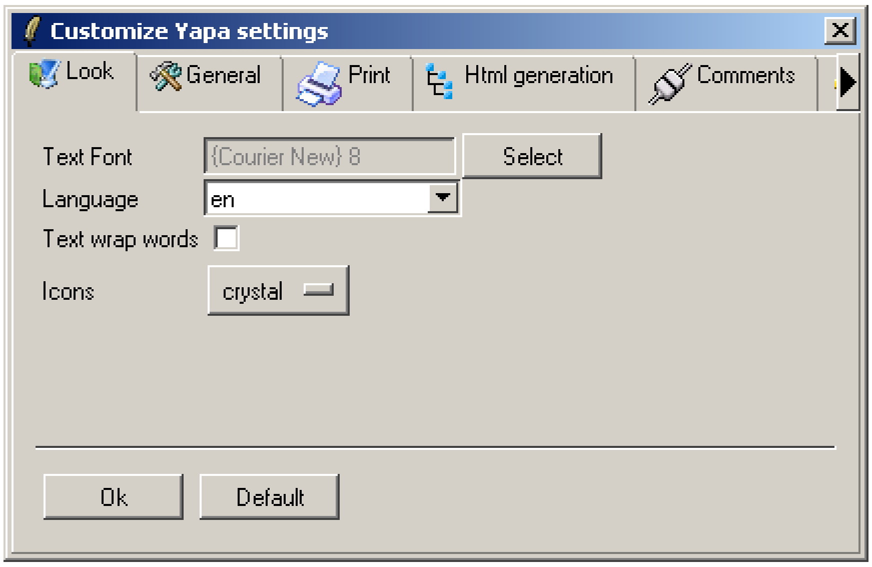
The bottom of the window is a terminal. Results of the search in files, and of the execution of a command are displayed in this terminal. The content of each terminal tab may be deleted, hidden, or saved into a log file for future analysing.

5.2.5 Recent files menu



When a file is opened, its path is stored in a list. If the user exits from Yapakit and startup Yapakit again, the same list is displayed into the ”Recent files” menu.

5.3 Customization

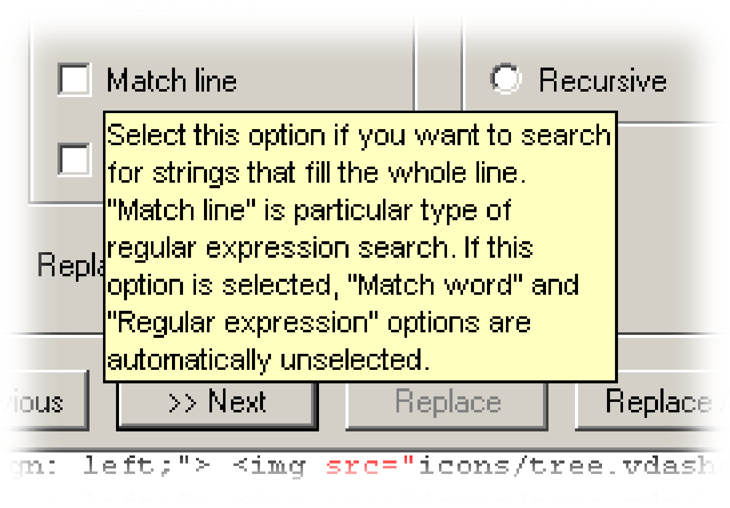


Text Font {Courier New} 8
Language en
Text wrap words ☐
Icons crystal

Customization thanks to the user interface : colors, fonts, mode associated with a given extension, indentation, comment/uncomment, insertion.

5.4 Internationalisation

This software is currently available in english and french. Any other foreign language can be added with a little effort by adding a ".msg" file in the "lang" directory. In the end, this help should be translated also...



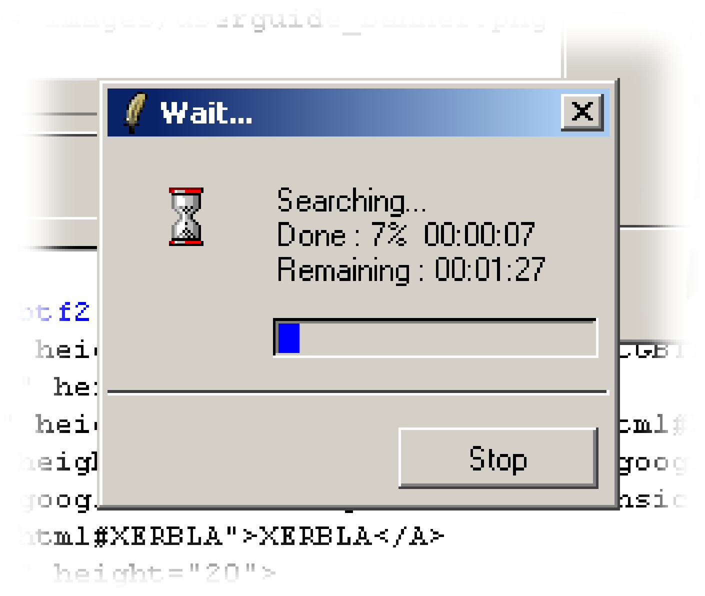
Select this option if you want to search for strings that fill the whole line. "Match line" is particular type of regular expression search. If this option is selected, "Match word" and "Regular expression" options are automatically unselected.

```
m: left;"> <img src="icons/tree.vdash
```

5.5 Help

This help is not very detailed because I suppose that the software is clear and simple enough so that any user can feel confortable with a few minutes. In addition, a significant effort has been made to provide contextual help. My idea was to put the help as close as possible to the menus and buttons. Balloon helps are displayed if the pointer (i.e. the mouse) stays quiet on any button of the toolbar. All menus are associated with help messages displayed in the status bar. All dialog boxes are kept as simple as possible and are explained, when necessary, with dynamic balloon messages.

5.6 Yapakit state



```
Searching...
Done : 7% 00:00:07
Remaining : 00:01:27
```

Allways know in what state Yapakit is.

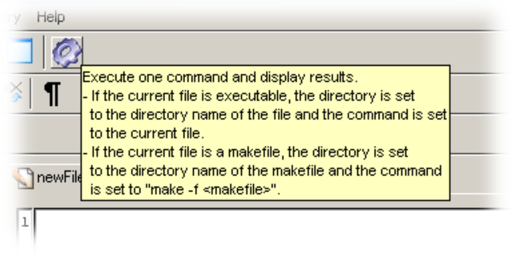
We use progress bars when possible and ”wait” dialogs when not possible. If a progress bar is displayed, the remaining time is available so that the user never loose time. When possible, a stop button is displayed so that the user can interrupt the current process and stop to loose time.

5.7 Automated tasks

The automated tasks allow to save a lot of time when processing several files : save all files, undo all, undo all files and many more. Do not make boring tasks. Time is precious. The saved time can therefore be devoted to more interesting tasks like playing tetris, debugging an algorithm, taking a coffee with colleagues, or going home, if possible.

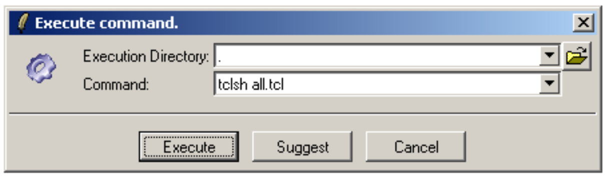
Chapter 6

Execute Command



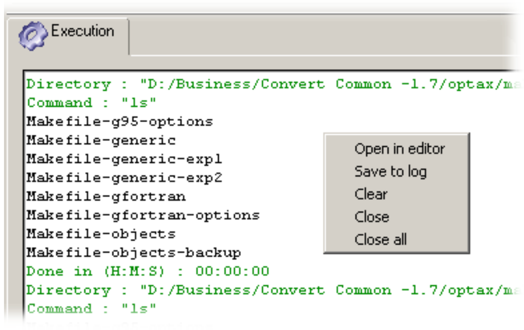
Execute one command and display results.
- If the current file is executable, the directory is set to the directory name of the file and the command is set to the current file.
- If the current file is a makefile, the directory is set to the directory name of the makefile and the command is set to "make -f <makefile>".

This dialog box allows to execute an external command. The parameters of the command are the directory name in which the command is executed and the name of the executable.



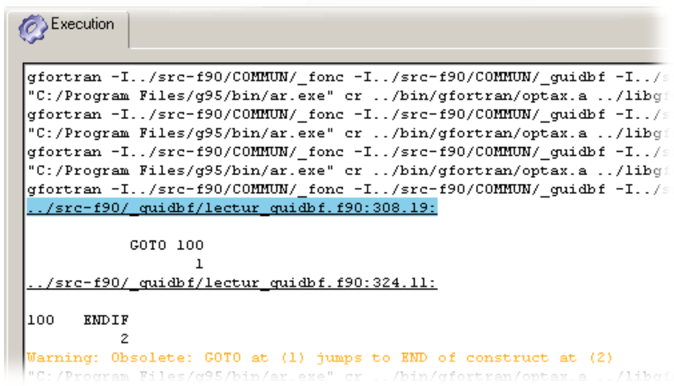
Execution Directory: .
Command: tclsh all.tcl

The suggest button allows to compute automatically the execution directory and the command in a given set of circumstances. If the text currently displayed is an executable script, the directory is set to the current directory and the command is set to the script. If the current file is a Tcl script, the same thing happens. The execution directory and the command parameters are associated with an history.



```
Directory : "D:/Business/Convert Common -1.7/optax/m
Command : "ls"
Makefile-g95-options
Makefile-generic
Makefile-generic-expl
Makefile-generic-exp2
Makefile-gfortran
Makefile-gfortran-options
Makefile-objects
Makefile-objects-backup
Done in (H:M:S) : 00:00:00
Directory : "D:/Business/Convert Common -1.7/optax/m
Command : "ls"
```

The result of the command is displayed into the Execution terminal, which content can be edited, saved or cleared. During the execution, the user can stop the process by clicking the ”Stop” button.



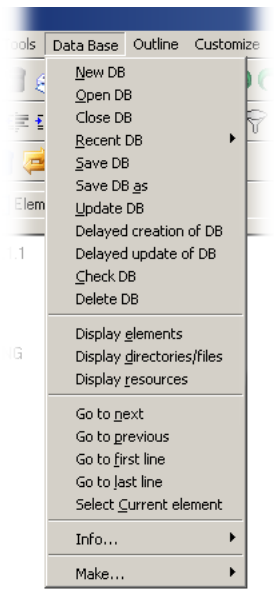
```
gfortran -I../src-f90/COMMUN/_fonc -I../src-f90/COMMUN/_guidbf -I../
"C:/Program Files/g95/bin/ar.exe" cr ../bin/gfortran/optax.a ../libg
gfortran -I../src-f90/COMMUN/_fonc -I../src-f90/COMMUN/_guidbf -I../
"C:/Program Files/g95/bin/ar.exe" cr ../bin/gfortran/optax.a ../libg
gfortran -I../src-f90/COMMUN/_fonc -I../src-f90/COMMUN/_guidbf -I../
"C:/Program Files/g95/bin/ar.exe" cr ../bin/gfortran/optax.a ../libg
gfortran -I../src-f90/COMMUN/_fonc -I../src-f90/COMMUN/_guidbf -I../
../src-f90/_guidbf/lectur_guidbf.f90:308.19:

           GOTO 100
                  1
../src-f90/_guidbf/lectur_guidbf.f90:324.11:

100   ENDIF
          2
Warning: Obsolete: GOTO at (1) jumps to END of construct at (2)
"C:/Program Files/g95/bin/ar.exe" cr ../bin/gfortran/optax.a ../libg
```

Run a makefile with the given arguments. Make is associated with special tags which allow to open the file and set the cursor to the correct error or warning line location immediately. Parsed messages for the following fortran compilers : gfortran, g95, intel fortran 7.1 and 8.1, pgi 5.1, Absoft.

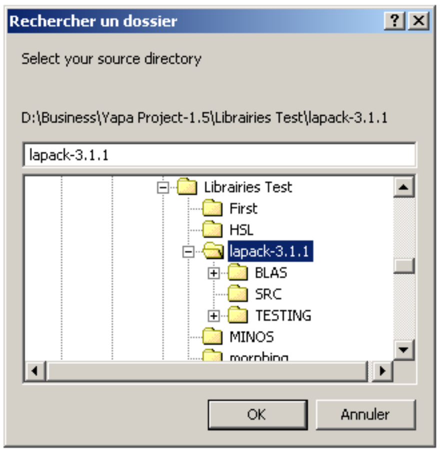
Chapter 7

Database Management

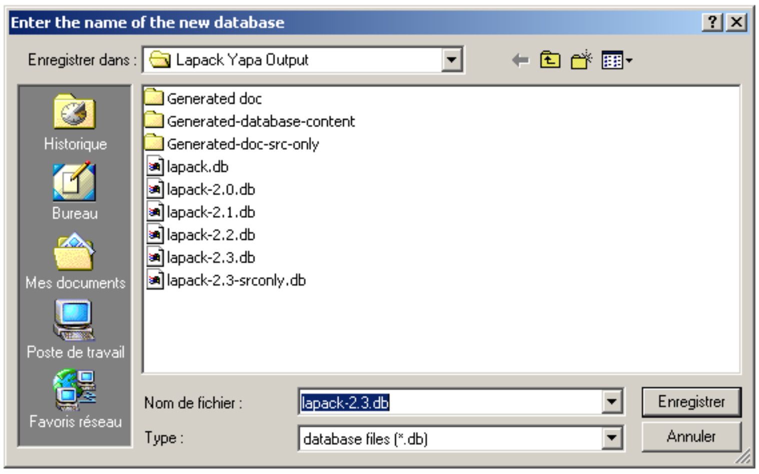
7.1 How to create a database ?



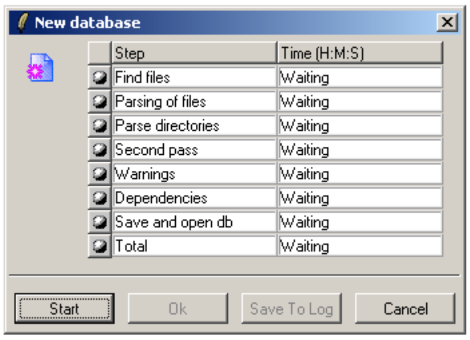
If the user has a set of fortran files in which he wants to navigate, he can create a database file which will allow to browse into the directories, files, elements (subroutine, function, etc...). The first step is to create a database file with the "New db" command.



The editor ask then the user to select a directory containing source code files. This directory is typically the "src" directory of the project. All parsable files in the selected directory and, recursively, in the sub-directories, will be stored into the database. Fortran 90 and fortran 77 files can appear in the same project.

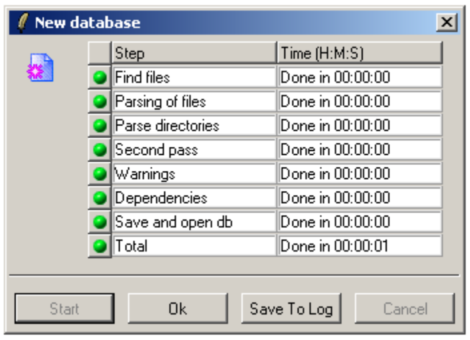


The user is then asked to enter the name of a database file. The editor suggests a name, which is the name of the last database opened. The file extension is ".db".


New database

Step	Time (H:M:S)
Find files	Waiting
Parsing of files	Waiting
Parse directories	Waiting
Second pass	Waiting
Warnings	Waiting
Dependencies	Waiting
Save and open db	Waiting
Total	Waiting

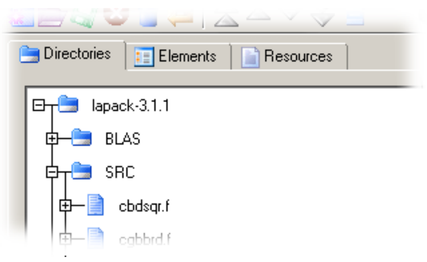
The ”New database” dialog is made of lines where each line is associated a step. If the user push the start button, each step will be processed one after another until the process is complete. The se steps correspond to the main steps to create the database. The longest steps are the parsing of the files, the second pass and the warnings computation : these steps are associated with a progress bar so that the user can interrupt the process anytime he wants if he thinks that it is too long.


New database

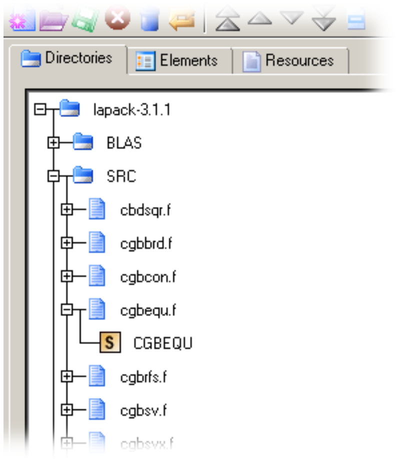
Step	Time (H:M:S)
Find files	Done in 00:00:00
Parsing of files	Done in 00:00:00
Parse directories	Done in 00:00:00
Second pass	Done in 00:00:00
Warnings	Done in 00:00:00
Dependencies	Done in 00:00:00
Save and open db	Done in 00:00:00
Total	Done in 00:00:01

When the process is done, all the times are summarized into the dialog box. These informations can be saved into a log file.

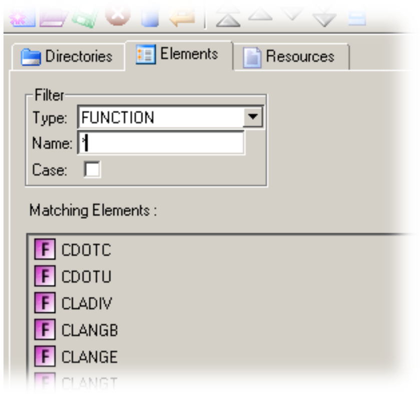
7.2 How to use a database



After that the database has been created, the database is automatically opened. The browse pane is filled with three tabs : Directories, Elements and Resources.

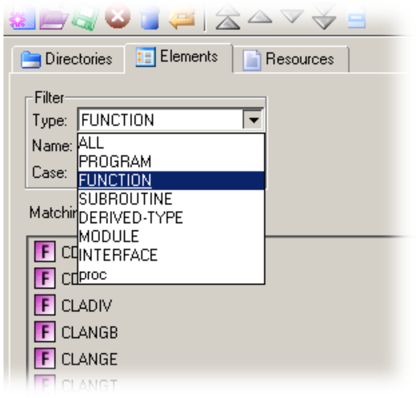


The "Directories" widget contains a tree with an explorer allowing the browse for directories and files.

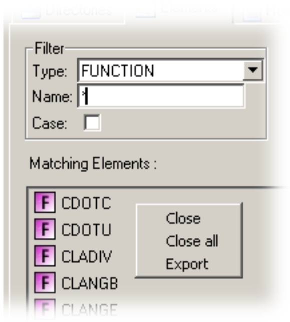


Type: FUNCTION
Name:
Case: ☐

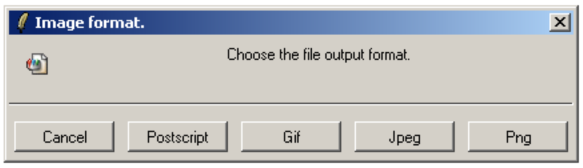
The "Elements" widget contains a list to display the elements of the project : subroutine, function, etc... The user can type a name pattern in the entry. By default, the entry appears empty, for performance reasons. If the pattern types "*", the list is updated with all the matching elements. The user can in fact any string pattern that the elements must match. For example, the string "c" would correspond with the list of all elements starting with the "c" or "C" letter. This filter can take the case into account. That is to say that, if the "Case" checkbox is ON and that the "c" letter is in the entry, then the elements starting with the downcase "c" letters will match, but not the elements starting with the upcase "C" letter. Note that the list is updated everytime the user enters letters into the entry.



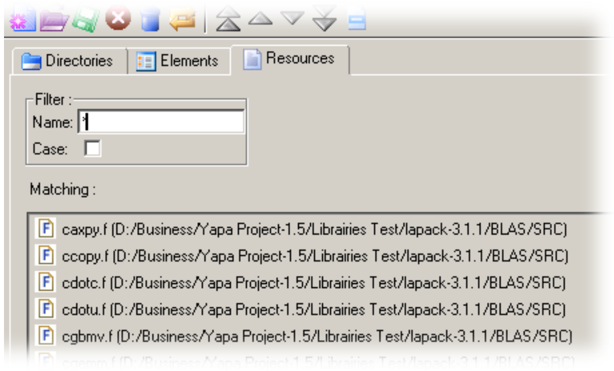
The user can select the type of the elements searched : PROGRAM, FUNCTION, SUBROUTINE, etc...



If the user right-click with the mouse located in the ”Elements” pane, a popup appears with 3 commands. The ”Close” command allows to close the current pane. The ”Close all” command allows to close all the panes at the same time. The ”Export” command allows to export the content of the pane into an image file.

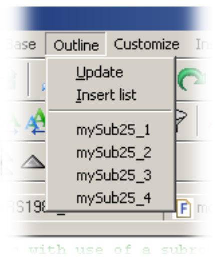


The user can select the file format between Postscript, Gif, Jpeg or Png so that a report can be easily filled with pictures generated by the editor.

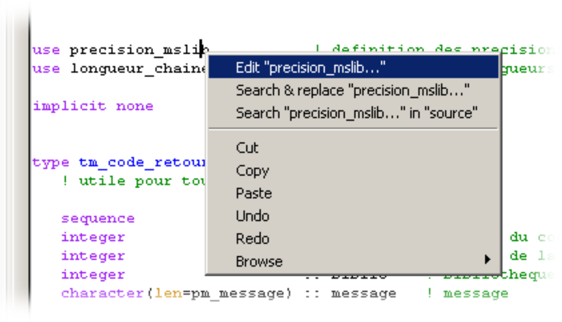


caxpy.f (D:/Business/Yapa Project-1.5/Librairies Test/lapack-3.1.1/BLAS/SRC)
ccopy.f (D:/Business/Yapa Project-1.5/Librairies Test/lapack-3.1.1/BLAS/SRC)
cdotc.f (D:/Business/Yapa Project-1.5/Librairies Test/lapack-3.1.1/BLAS/SRC)
cdotu.f (D:/Business/Yapa Project-1.5/Librairies Test/lapack-3.1.1/BLAS/SRC)
cgbmv.f (D:/Business/Yapa Project-1.5/Librairies Test/lapack-3.1.1/BLAS/SRC)

The ”Resources” allows to display the list of files of the project. The entry allows to define the file pattern. The ”Case” checkbox allows to set the file pattern that the files must match.

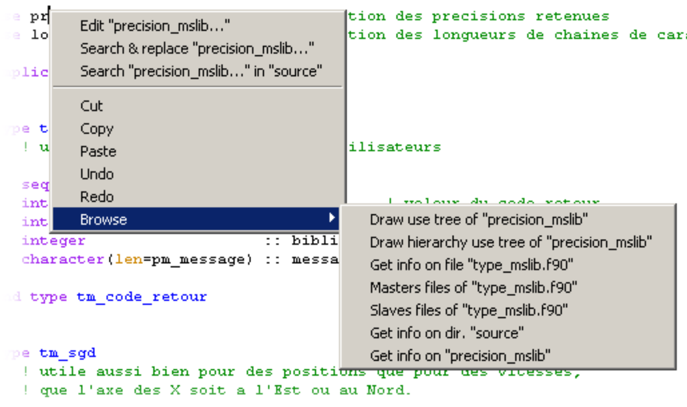


When a file from the database is opened, the ”Outline” menu contains the elements found in the file. When the user clicks in the menu, the cursor is set at the line corresponding to the starting of the definition of the element.

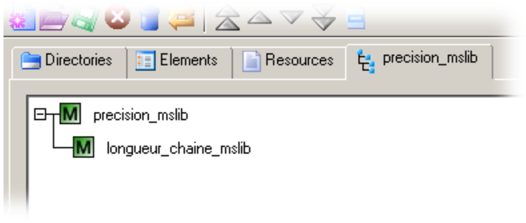

Edit "precision_mslib..."
Search & replace "precision_mslib..."
Search "precision_mslib..." in "source"
Cut
Copy
Paste
Undo
Redo
Browse

```
use precision_mslib
use longueur_chaine

implicit none


type tm_code_retour
   ! utile pour tou

   sequence
   integer
   integer
   integer
   character(len=pm_message) :: message   ! message
```

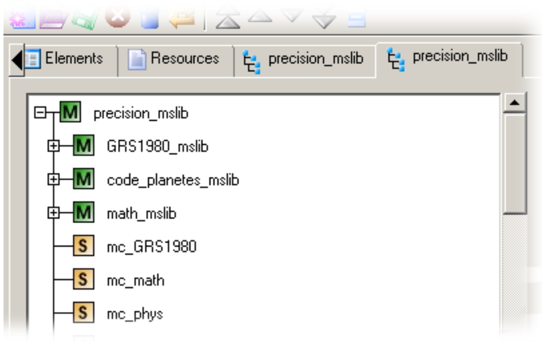
When editing a file in the project, the user can right-click on an element which is in the database. In the example shown above, the user right-click on the string ”precision_mslib” which is a module used by the current module. The popup windows which appears allows to edit the module ”precision_mslib”, which means that the file containing the module ”precision_mslib” will be opened (if it is not allready opened), then the cursor will be set at the starting of the definition of the module ”precision_mslib”. This allows a fast navigation into the project. But there is more features, stored in the ”Browse” sub-menu of the popup window.


Edit "precision_mslib..."
Search & replace "precision_mslib..."
Search "precision_mslib..." in "source"
Cut
Copy
Paste
Undo
Redo
Browse
Draw use tree of "precision_mslib"
Draw hierarchy use tree of "precision_mslib"
Get info on file "type_mslib.f90"
Masters files of "type_mslib.f90"
Slaves files of "type_mslib.f90"
Get info on dir. "source"
Get info on "precision_mslib"

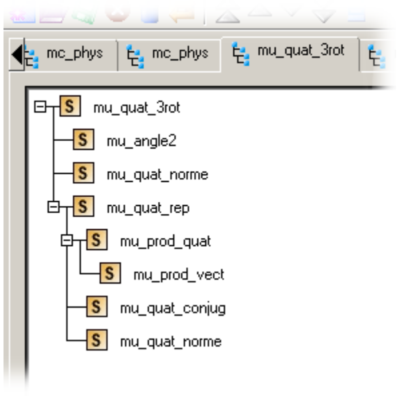
The "Browse" sub-menu allows to use extensively the content of the database. One can draw the "Use tree" of a fortran 90 module, the call tree of a fortran 77 or 90 subroutine, function or program. One can draw the masters or the slaves of a given file (the masters of a file are the files that the current file depends on, in the dependency tree). One can finally display informations on the current element, file or directory.



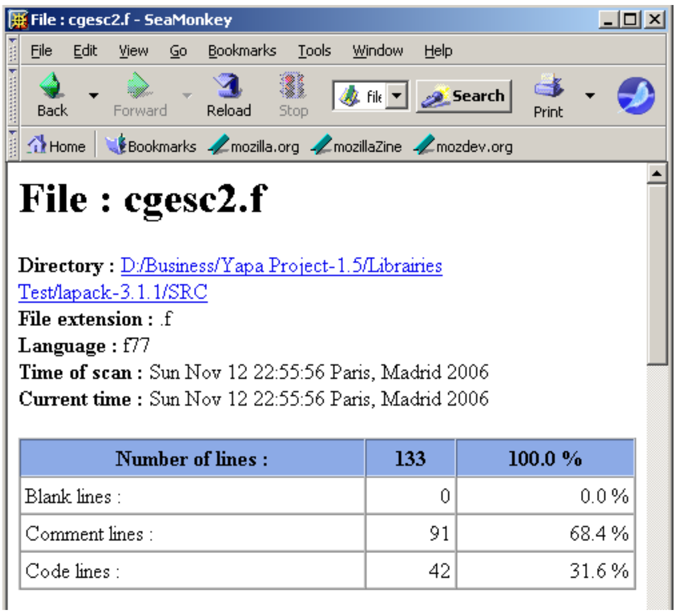
In the example above, one can draw the "Use tree" of the module "precision_mslib". This tree has module "precision_mslib" as root and all used modules of the module "precision_mslib" as children. The depth of the tree is not limited. This is a fortran 90 feature.



The hierarchy use tree of the module "precision_mslib" is made of all the elements which use the module "precision_mslib". For example, the subroutine "mc_GRS1980" is using the module "precision_mslib". Again, there is no limit in the depth of that tree. The "Use tree" feature is very useful, but the "Hierarchy use tree" is extremely useful when analysing the structure of a fortran 90 software. This is a fortran 90 feature.



The "Call tree" of any subroutine, function, program or interface can be drawn. This is a fortran 77 and fortran 90 feature (except that the "interface" keyword is only defined in fortran 90). Notice that fortran 77 and fortran 90 elements can appear in the same call tree, since no difference is made between the two types of elements.

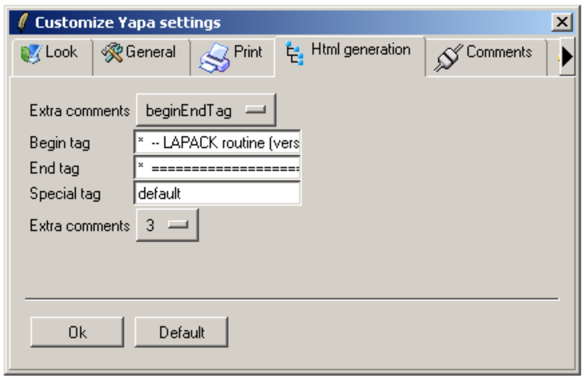

File : cgesc2.f - SeaMonkey
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop Search Print
Home Bookmarks mozilla.org mozillaZine mozdev.org

File : cgesc2.f

Directory : D:/Business/Yapa Project-1.5/Libraries Test/lapack-3.1.1/SRC
File extension : .f
Language : f77
Time of scan : Sun Nov 12 22:55:56 Paris, Madrid 2006
Current time : Sun Nov 12 22:55:56 Paris, Madrid 2006

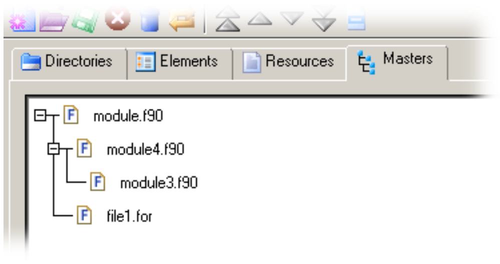
Number of lines :	133	100.0 %
Blank lines :	0	0.0 %
Comment lines :	91	68.4 %
Code lines :	42	31.6 %

Generates an html documentation of the project stored in memory. This documentation is filled with the elements (Directories, Files and Elements like Subroutines, Functions, Modules, etc...). It allows the user to read statistics abouts the project (number of files, number of lines and many more).

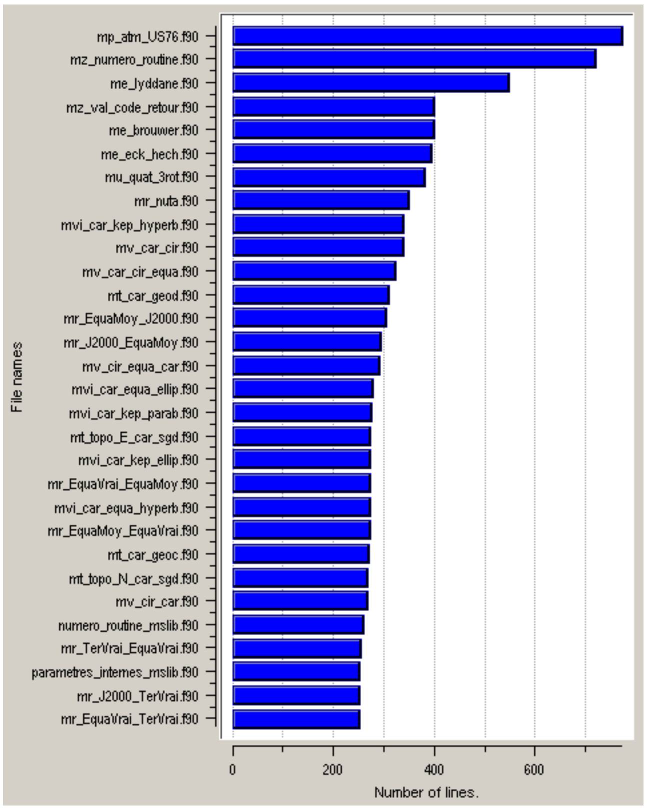


Extra comments: beginEndTag
Begin tag: * -- LAPACK routine (vers
End tag: * =================
Special tag: default
Extra comments: 3

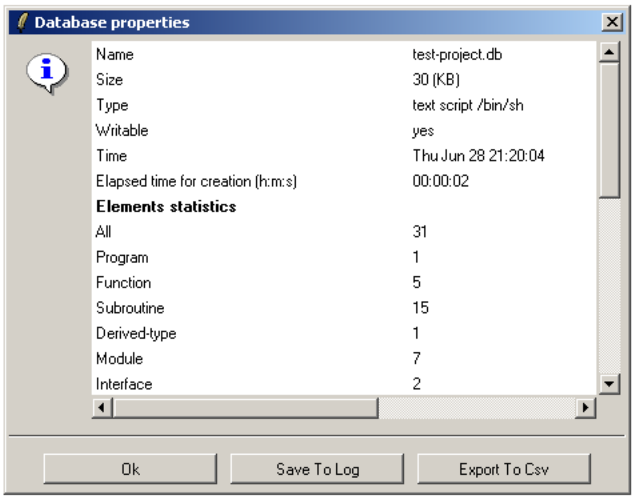
The html documentation displays the comments extracted from the source code and associates them to the correct element. The "Customize" menu contains the sub-menu "Settings", in which the user can find the tab "Html generation". This allows the user to define how to extract the comments from the fortran source code.



Explore the file dependecies of your fortran files. Display the masters or the slaves of a file : the masters of a given file are those that the current file is depending on. The dependencies are based on the call tree (fortran 77 or fortran 90) and on "use" statements in fortran 90.

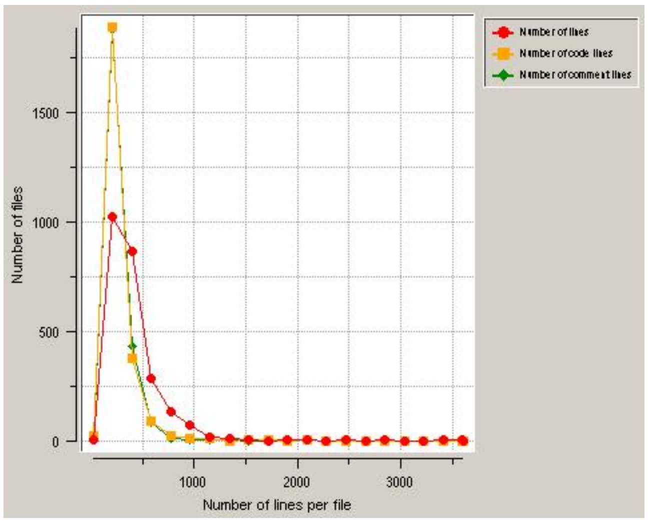


Display statistics of the project. Display list of 30 largest files in the project.

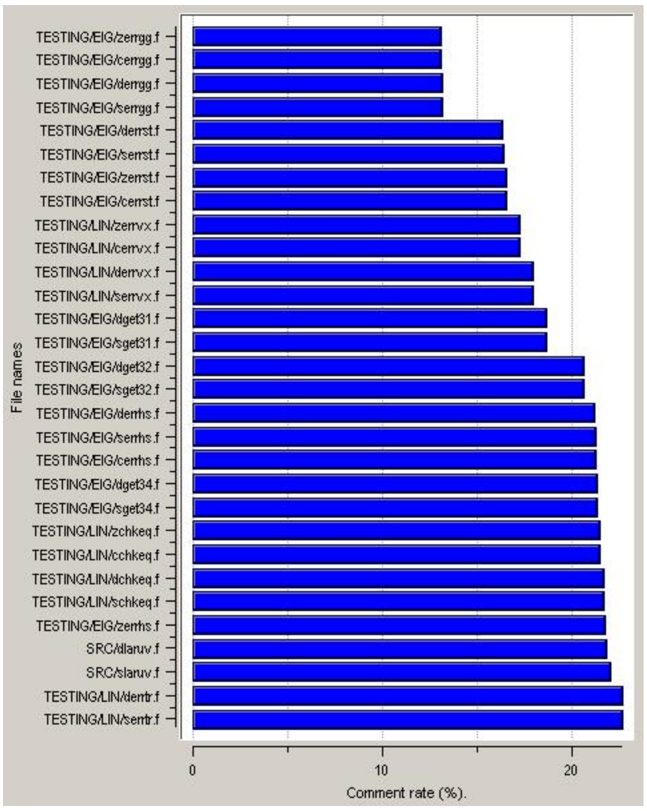


Name	test-project.db
Size	30 (KB)
Type	text script /bin/sh
Writable	yes
Time	Thu Jun 28 21:20:04
Elapsed time for creation (h:m:s)	00:00:02
Elements statistics	
All	31
Program	1
Function	5
Subroutine	15
Derived-type	1
Module	7
Interface	2

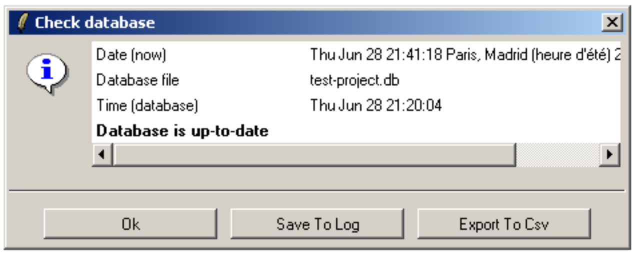
Display the database properties (number of directories, of files, of elements, the date of the database, etc...), the file properties (number of files, code rate, comment rate, etc...).



Displays the lines distribution as a graph (for example the comment rate distribution).



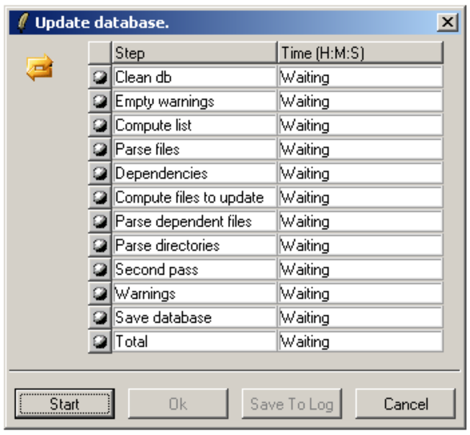
Display the 30 most commented files as a bar chart, the 30 least commented files (bar chart also).



Date (now)	Thu Jun 28 21:41:18 Paris, Madrid (heure d'été) 2
Database file	test-project.db
Time (database)	Thu Jun 28 21:20:04

Database is up-to-date

Allow to know whether a given database file is up-to-date or not thanks to the ”Check database” command.



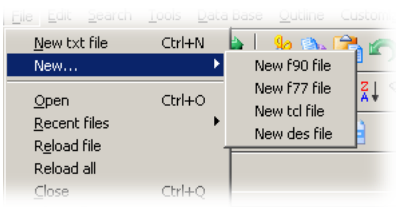
Step	Time (H:M:S)
Clean db	Waiting
Empty warnings	Waiting
Compute list	Waiting
Parse files	Waiting
Dependencies	Waiting
Compute files to update	Waiting
Parse dependent files	Waiting
Parse directories	Waiting
Second pass	Waiting
Warnings	Waiting
Save database	Waiting
Total	Waiting

Update a database with minimal processing. When source code files are modified, the informations stored in the database are not up-to-date anymore. The database needs an update, which can be done by this dialog box. The database update is minimal in the sense that it does not need to parse the whole project, but only the files which have been modified (or created), and the files that depends on that modified files.

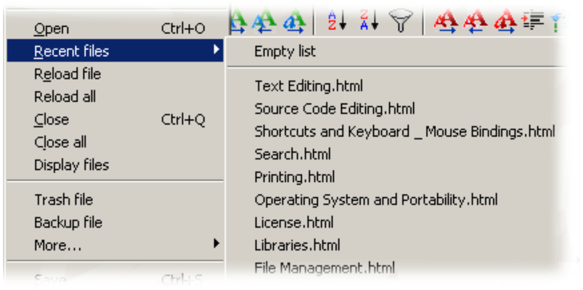
Chapter 8

File Management

8.1 New file

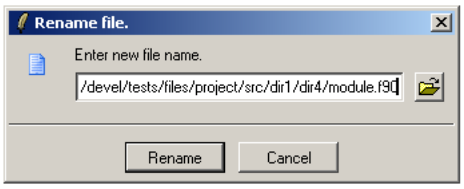


New files can be created with the correct extension and mode (.f90, .txt, etc...)



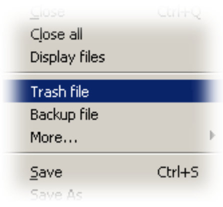
Recent files are listed in the file menu and can be opened from one session to another.

8.2 Renaming a file

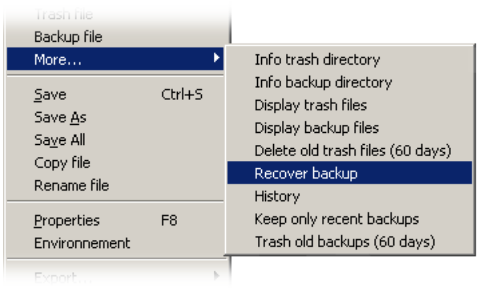


/devel/tests/files/project/src/dir1/dir4/module.f90

8.3 Files are not deleted



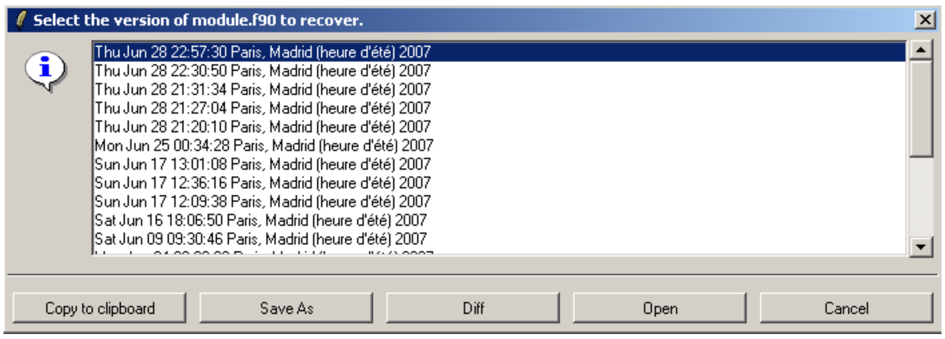
Never delete a file : they are instead put in the trash directory. In the trash directory, the files are managed so that they are never deleted, but renamed instead (for example, if you trash two files with the same file tail). In fact, in the editor, there is no way to directly delete a file from the disk. Instead, the editor allows to automatically remove files which are older than 60 days. In fact, this cannot lead to space problems, because files generally edited by hand are small in size relatively to the space available on modern disks.



Files are automatically backuped wherever the user opens, save and quit a file. Backup files are named with their date and can be recoverd if necessary.

The backup directory is managed : if the disk space is limited, the user can keep a limited number of versions for each original file or can delete backup files older than 1 month.

8.4 File history



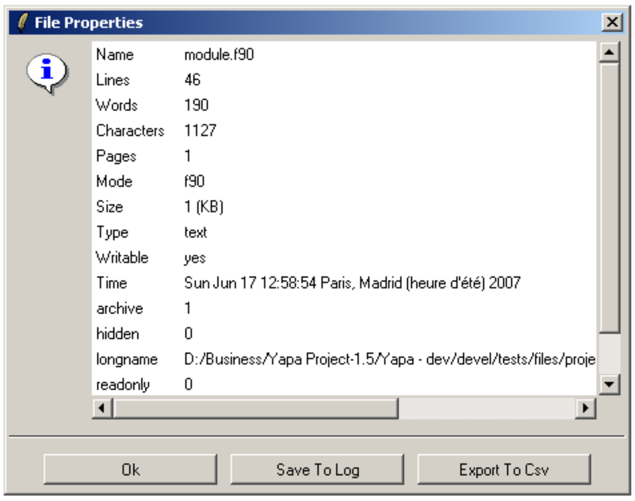
```
Thu Jun 28 22:57:30 Paris, Madrid (heure d'été) 2007
Thu Jun 28 22:30:50 Paris, Madrid (heure d'été) 2007
Thu Jun 28 21:31:34 Paris, Madrid (heure d'été) 2007
Thu Jun 28 21:27:04 Paris, Madrid (heure d'été) 2007
Thu Jun 28 21:20:10 Paris, Madrid (heure d'été) 2007
Mon Jun 25 00:34:28 Paris, Madrid (heure d'été) 2007
Sun Jun 17 13:01:08 Paris, Madrid (heure d'été) 2007
Sun Jun 17 12:36:16 Paris, Madrid (heure d'été) 2007
Sun Jun 17 12:09:38 Paris, Madrid (heure d'été) 2007
Sat Jun 16 18:06:50 Paris, Madrid (heure d'été) 2007
Sat Jun 09 09:30:46 Paris, Madrid (heure d'été) 2007
```

The file history of the current file can be traced thanks to the same feature (from the File > More > History menu). The ”Diff” button allows to display the differences between the current version and the selected backup version. It is based on the Tkdiff external tool.

8.5 Save all files

If several files are modified, do not waste your time to save one file after another : save all files in one click.

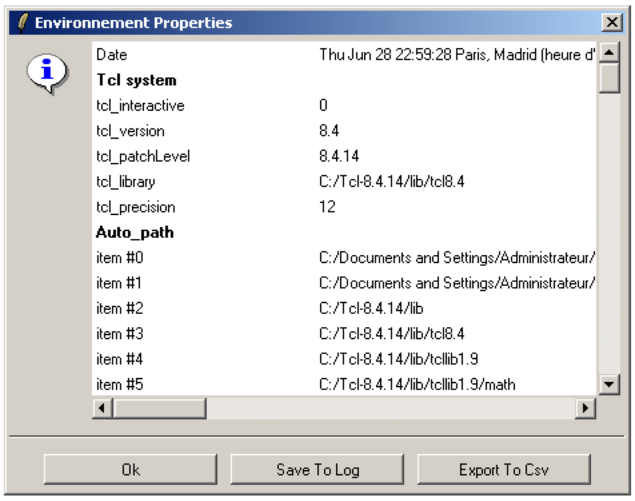
8.6 Files properties



Name	module.f90
Lines	46
Words	190
Characters	1127
Pages	1
Mode	f90
Size	1 (KB)
Type	text
Writable	yes
Time	Sun Jun 17 12:58:54 Paris, Madrid (heure d'été) 2007
archive	1
hidden	0
longname	D:/Business/Yapa Project-1.5/Yapa - dev/devel/tests/files/proje
readonly	0

Get information about the file : date, size, number of lines, words, mode, size, etc....

8.7 Environment properties



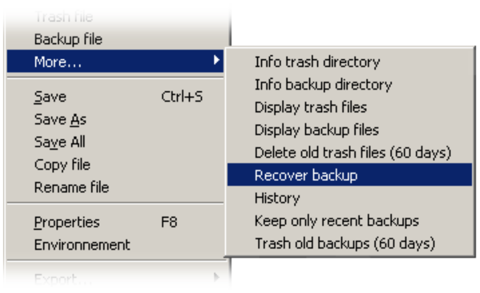
Date	Thu Jun 28 22:59:28 Paris, Madrid (heure d'
Tcl system	
tcl_interactive	0
tcl_version	8.4
tcl_patchLevel	8.4.14
tcl_library	C:/Tcl-8.4.14/lib/tcl8.4
tcl_precision	12
Auto_path	
item #0	C:/Documents and Settings/Administrateur/
item #1	C:/Documents and Settings/Administrateur/
item #2	C:/Tcl-8.4.14/lib
item #3	C:/Tcl-8.4.14/lib/tcl8.4
item #4	C:/Tcl-8.4.14/lib/tcllib1.9
item #5	C:/Tcl-8.4.14/lib/tcllib1.9/math

Display the environnment : Tcl system, operating system, environment variables, disk volumes. The Tcl "::auto_path" global variable is a list of directories in which the Tcl system automatically searches for scripts and packages.

Chapter 9

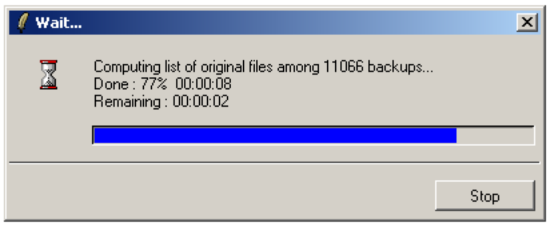
How to recover a backup ?

9.1 Automatic backup



Files are automatically backuped whenever they are opened, saved and closed. The ”Recover backup” feature in the ”File” menu allows to recover any backuped file.

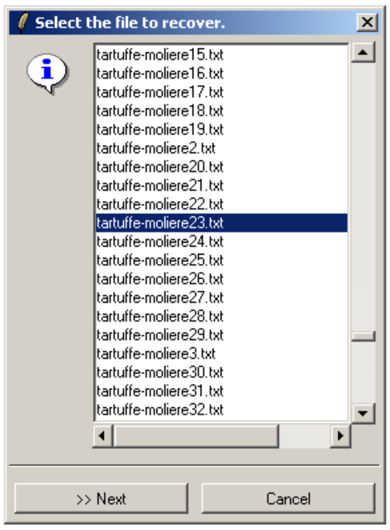
9.2 Recover a backup



Computing list of original files among 11066 backups...
Done : 77% 00:00:08
Remaining : 00:00:02

The command behind fhe ”File > More ... > Recover backup” allows to see and get all backups associated with a given file. When the user wants to recover a backup, the software computes the list of original files for which backups are available.

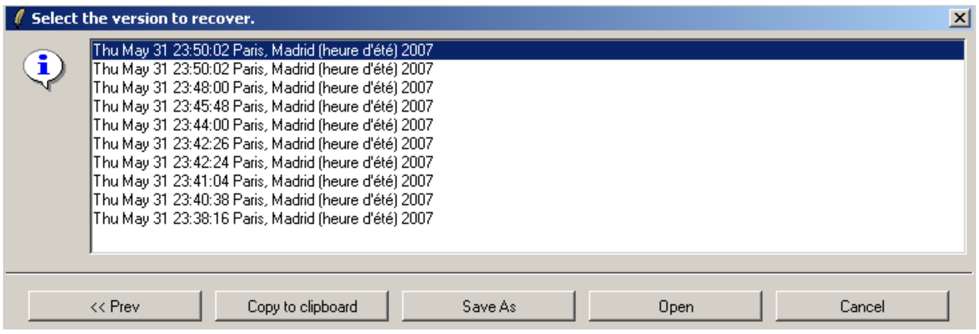
9.3 Select the original to recover



tartuffe-moliere15.txt
tartuffe-moliere16.txt
tartuffe-moliere17.txt
tartuffe-moliere18.txt
tartuffe-moliere19.txt
tartuffe-moliere2.txt
tartuffe-moliere20.txt
tartuffe-moliere21.txt
tartuffe-moliere22.txt
tartuffe-moliere23.txt
tartuffe-moliere24.txt
tartuffe-moliere25.txt
tartuffe-moliere26.txt
tartuffe-moliere27.txt
tartuffe-moliere28.txt
tartuffe-moliere29.txt
tartuffe-moliere3.txt
tartuffe-moliere30.txt
tartuffe-moliere31.txt
tartuffe-moliere32.txt

The user then select the original file to recover in a list box and clicks ”Next”.

9.4 Select the version



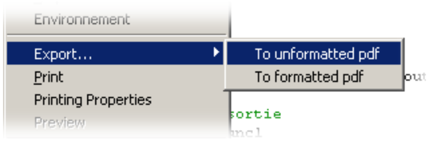
Thu May 31 23:50:02 Paris, Madrid (heure d'été) 2007
Thu May 31 23:50:02 Paris, Madrid (heure d'été) 2007
Thu May 31 23:48:00 Paris, Madrid (heure d'été) 2007
Thu May 31 23:45:48 Paris, Madrid (heure d'été) 2007
Thu May 31 23:44:00 Paris, Madrid (heure d'été) 2007
Thu May 31 23:42:26 Paris, Madrid (heure d'été) 2007
Thu May 31 23:42:24 Paris, Madrid (heure d'été) 2007
Thu May 31 23:41:04 Paris, Madrid (heure d'été) 2007
Thu May 31 23:40:38 Paris, Madrid (heure d'été) 2007
Thu May 31 23:38:16 Paris, Madrid (heure d'été) 2007

The dialog then displays a list of backup versions available for the original file. The content can be copied to clipboard (for direct insertion in another file), saved as another file or directly opened.

Chapter 10

Printing

10.1 Export to pdf

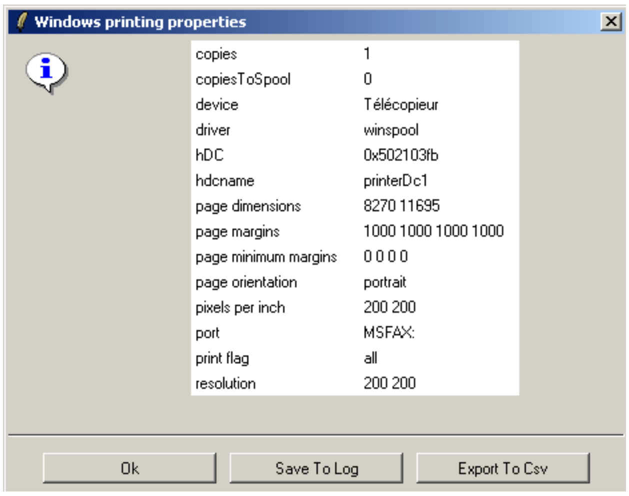


The user can export the source code into formatted or unformatted pdf files. Unformatted pdf files are simple pdf with no special layout. In formatted pdf files, the editor uses a different font for comments, for keywords, for standard library functions and for strings.

10.2 Printing with Unix / Linux

Under the Unix systems, one can visulalize the file before printing it thanks to a customized postscript viewer (”gv” for instance). Under unix, the printing system is based on the a2ps and lpr commands.

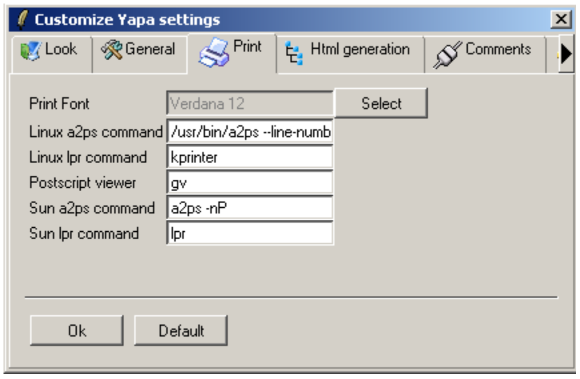
10.3 Printing with Windows



copies	1
copiesToSpool	0
device	Télécopieur
driver	winspool
hDC	0x502103fb
hdcname	printerDc1
page dimensions	8270 11695
page margins	1000 1000 1000 1000
page minimum margins	0 0 0 0
page orientation	portrait
pixels per inch	200 200
port	MSFAX:
print flag	all
resolution	200 200

Under Windows, the printing system is integrated into Yapakit.

10.4 Customize printing settings



Print Font	Verdana 12
Linux a2ps command	/usr/bin/a2ps --line-numb
Linux lpr command	kprinter
Postscript viewer	gv
Sun a2ps command	a2ps -nP
Sun lpr command	lpr

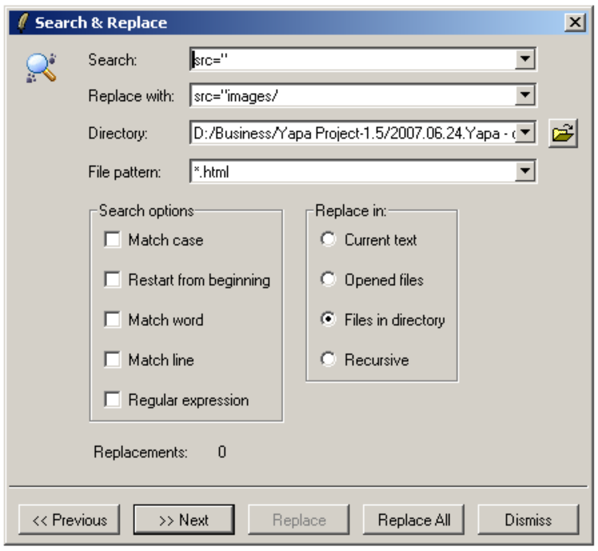
The user can customize the a2ps and the gv command for printing thanks to the ”Customize/Settings/Print” menu.

Chapter 11

Search and Replace

The search feature is a very important tool for any editor. As such, it has been a domain of many developments, is rich of many powerful features and has suffered many tests. The main goal is to provide a tool which can be used for the simplest search task of searching in the current text as well as the more complex task of replacing a word into a whole set of files, located in many different directories. The current dialog allows the user to master the process of replacing a string in several files, with the possibility of interrupting the process when desired.

11.1 Search & Replace Dialog box



Search & Replace

Search: src="

Replace with: src="images/

Directory: D:/Business/Yapa Project-1.5/2007.06.24.Yapa - (

File pattern: *.html

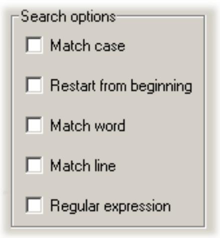
Search options
- ☐ Match case
- ☐ Restart from beginning
- ☐ Match word
- ☐ Match line
- ☐ Regular expression

Replace in:
- ○ Current text
- ○ Opened files
- ● Files in directory
- ○ Recursive

Replacements: 0

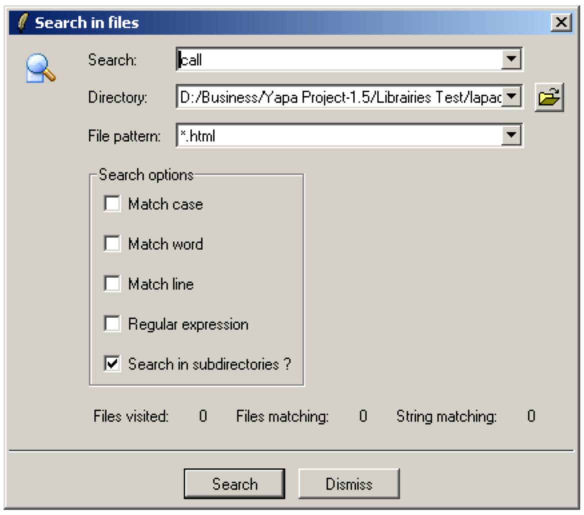
The ”Search & Replace” dialog is mainly interactive and allows to search for a string in the current text. If the ”Opened files” options is chosen, only files allready opened in the editor are processed. If the ”Files in directory” option is chosen, only files directly in the chosen directory are chosen. If ”Recursive” option is chosen, all files in the chosen directory, and recursively, in all subdirectories are processed. These search and replace algorithms (and especially the last one) are tailored so as to be efficient when large amount of data is processed. The ”Recursive” option, which is only partly interactive, has been optimized on set of files which contain more than 500 files : if one file is modified after a replacement is done, the new file content is saved and then closed so that the editor has only a limited number of opened files at one time.

11.2 Search options



Allow to take (or not to take) the case into account, to match a single word, a single line or even a regular expression.

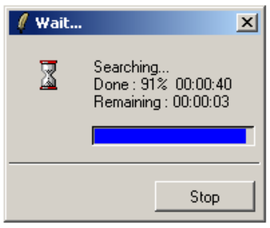
11.3 Search in files



Search: call
Directory: D:/Business/Yapa Project-1.5/Librairies Test/lapac
File pattern: *.html
Search options
☐ Match case
☐ Match word
☐ Match line
☐ Regular expression
☑ Search in subdirectories ?
Files visited: 0 Files matching: 0 String matching: 0

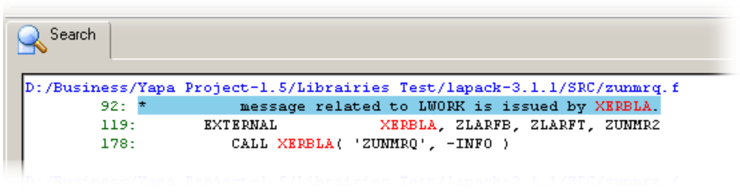
The ”Search in files” dialog, which is not interactive, allows to search for file contents on a larger file set that the ”Search & Replace” dialog. The results are displayed in the ”Search” terminal in the terminal pane, located at the bottom of the editor.

11.4 Search in files progress bar



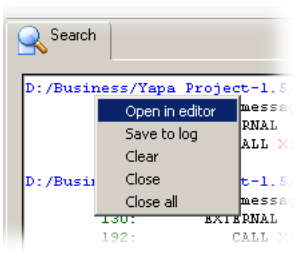
When the "search in files" process is running, a progress bar is displayed so that the user is informed of the current state of the search process and can stop it, if it is too long, an error has been made on the search, etc...

11.5 Search terminal



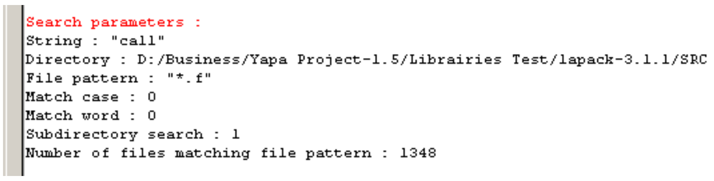
```
D:/Business/Yapa Project-1.5/Librairies Test/lapack-3.1.1/SRC/zunmrq.f
        92: *             message related to LWORK is issued by XERBLA.
       119:         EXTERNAL             XERBLA, ZLARFB, ZLARFT, ZUNMR2
       178:            CALL XERBLA( 'ZUNMRQ', -INFO )
```

The result is displayed in the interactive search terminal. The user only has to click in the search terminal to open the file and set the cursor at the corresponding found line. In the terminal, the string matching the pattern is displayed with a special color (here in red) so that the result of the search is obvious for the user.



The content of the search terminal can be opened in the editor as a regular plain new text, so that the user can modify the text, make new searches, filter the result, etc ... It can also be saved into a log file, so that it can be sent by email to a colleague, etc... The "Clear" command simply deletes the content of the terminal : all "Search in files" are filling that terminal, which content can become large after several searches are made.

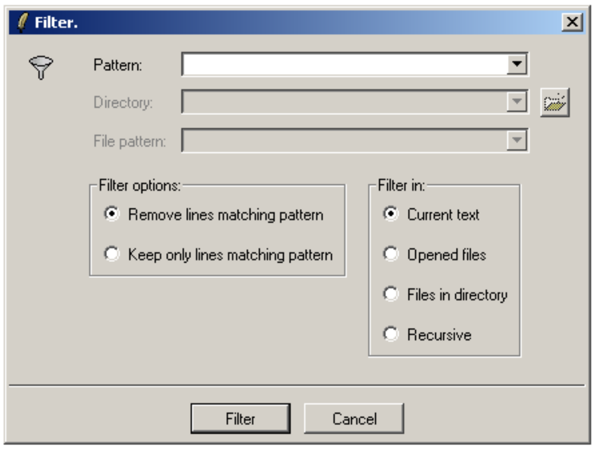
11.6 Search terminal popup



```
Search parameters :
String : "call"
Directory : D:/Business/Yapa Project-1.5/Librairies Test/lapack-3.1.1/SRC
File pattern : "*.f"
Match case : 0
Match word : 0
Subdirectory search : 1
Number of files matching file pattern : 1348
```

An abstract of the "Search in files" options is displayed in the terminal.

11.7 Filter



Filter options: (● Remove lines matching pattern) ○ Keep only lines matching pattern

Filter in: (● Current text) ○ Opened files ○ Files in directory ○ Recursive

Filter lines matching specific pattern in the current file, or in multiple files (opened files, files in directory, files in directory and in subdirectories).

Chapter 12

Editing Source Code

Several features are available to ease the editing of the Fortran source code. Syntax highlighting is available for Fortran 77 and Fortran 90 source code. Moreover, the automatic indentation and comment / uncomment features are provided in the editor.

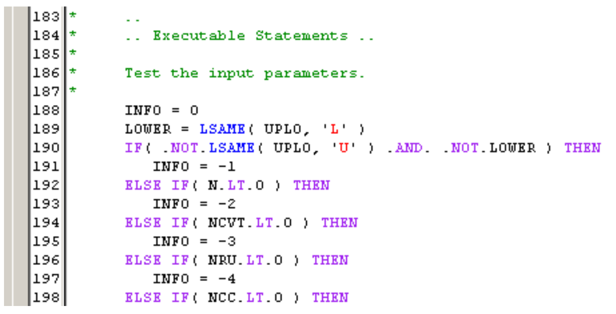
With the additionl plugins, syntax highlighting is provided for other languages as well : elisp, gnuplot, java, perl, python, shell.

12.1 Syntax highlighting

12.1.1 Syntax highlighting for fortran 77

The syntax highlighting feature of the fortran 77 is based on a complex algorithm which takes into account all the following particularities of the Fortran 77 language with different colors :

- fixed format and TAB-format are taken into account, with colored labels,
- comments are colored,
- pre-processing lines are colored,
- the 6-th column (continuation) is colored,
- strings between ör ćharacters are colored,
- keyword and standard library items are colored separately,
- everything after the 72-th column is colored.

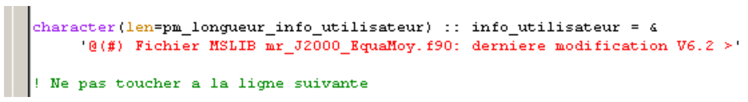


```
*       ..
*       .. Executable Statements ..
*
*       Test the input parameters.
*
        INFO = 0
        LOWER = LSAME( UPLO, 'L' )
        IF( .NOT.LSAME( UPLO, 'U' ) .AND. .NOT.LOWER ) THEN
           INFO = -1
        ELSE IF( N.LT.0 ) THEN
           INFO = -2
        ELSE IF( NCVT.LT.0 ) THEN
           INFO = -3
        ELSE IF( NRU.LT.0 ) THEN
           INFO = -4
        ELSE IF( NCC.LT.0 ) THEN
```

12.1.2 Syntax highlighting for fortran 90

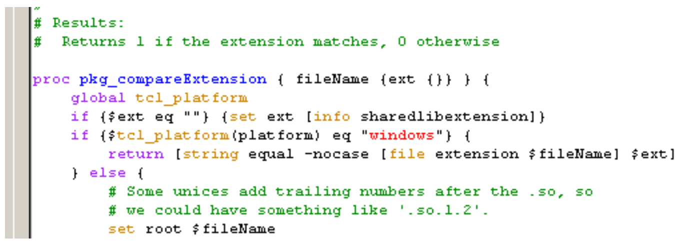
The syntax highlighting feature of the fortran 90 is based on a complex algorithm which takes into account all the following particularities of the Fortran 90 language with different colors :

- free format is taken into account,
- comment line and in-line comments are colored,
- pre-processing lines are colored,
- strings between ör ćharacters are colored,
- keyword and standard library items are colored separately,
- everything after the 132-th column is colored.



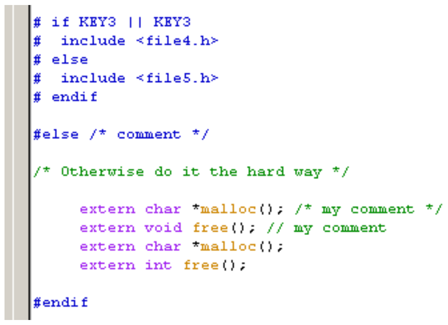
```
character(len=pm_longueur_info_utilisateur) :: info_utilisateur = &
     '@(#) Fichier MSLIB mr_J2000_EquaMoy.f90: derniere modification V6.2 >'

! Ne pas toucher a la ligne suivante
```

12.1.3 Syntax highlighting for Tcl



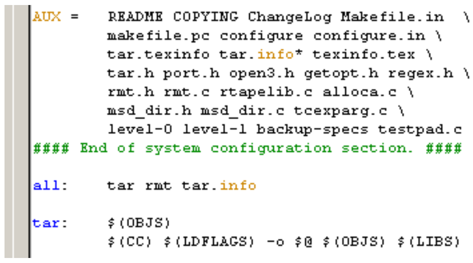
```
# Results:
#  Returns 1 if the extension matches, 0 otherwise

proc pkg_compareExtension { fileName {ext {}} } {
    global tcl_platform
    if {$ext eq ""} {set ext [info sharedlibextension]}
    if {$tcl_platform(platform) eq "windows"} {
        return [string equal -nocase [file extension $fileName] $ext]
    } else {
        # Some unices add trailing numbers after the .so, so
        # we could have something like '.so.1.2'.
        set root $fileName
```

12.1.4 Syntax highlighting for C/C++



```
# if KEY3 || KEY3
#  include <file4.h>
# else
#  include <file5.h>
# endif

#else /* comment */

/* Otherwise do it the hard way */

     extern char *malloc(); /* my comment */
     extern void free(); // my comment
     extern char *malloc();
     extern int free();

#endif
```

12.1.5 Syntax highlighting for Makefile



```
AUX =   README COPYING ChangeLog Makefile.in  \
        makefile.pc configure configure.in \
        tar.texinfo tar.info* texinfo.tex \
        tar.h port.h open3.h getopt.h regex.h \
        rmt.h rmt.c rtapelib.c alloca.c \
        msd_dir.h msd_dir.c tcexparg.c \
        level-0 level-1 backup-specs testpad.c
#### End of system configuration section. ####

all:    tar rmt tar.info

tar:    $(OBJS)
        $(CC) $(LDFLAGS) -o $@ $(OBJS) $(LIBS)
```

If the automatically selected mode is not correct, the user can manually highlight a source code with the mode he wants with the ”Tools > Hightlight current text” command.

12.2 Auto-indent fortran

Automatic indentation is provided for fortran 90 and fortran 77 source code.

The auto-indent feature of the fortran 77 takes into account all the following particularities of the Fortran 77 language :

- comment lines are never indented,
- pre-processing lines are taken into account,
- continuation lines are taken into account,
- labels are taken into account,
- source code blocks (function, subroutine, program, if, do, etc...) are indented.

The algorithm is based on a wuit complex algorithm which is heavily tested. A specific fortran 90 auto-indent algorithm is available.

Auto-indent is also available for Tcl and txt files.

Moreover, the following features are available for fortran source code :

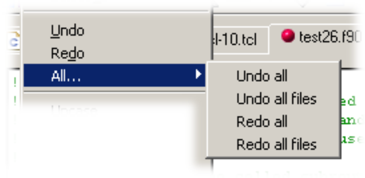
- Commenting/uncommenting source code.
- Upcase/downcase/capitalize keywords of a source code file.

Chapter 13

Text Editing

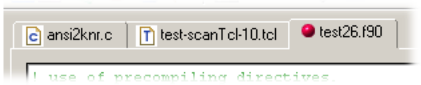
This section discuss the features which allow the user to process texts. Thanks to the rich set of features provided by the underlying langage on which the editor was built (Tcl/Tk), the editor provides many features to process texts. For example, the editor provides unlimited undo / redo, allow to justify texts or to upcase or downcase a selected region. These features are useful when writing simple text notes or when managing long data files.

13.1 Unlimited undo / redo



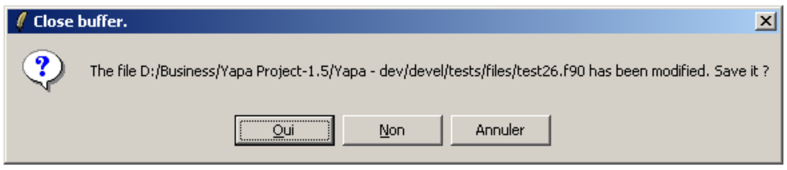
There is no limit in the undo/redo feature. One can also undo/redo all modifications in one text. Furthermore, on can undo/redo all modifications in all texts.

13.2 Modified file



When a text is modified, a red ball is displayed at the left of the name so it is easy to know what are all the unsaved texts.

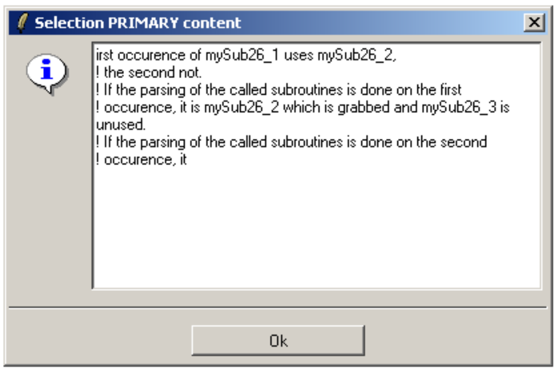
13.3 Quit with unsaved files



The file D:/Business/Yapa Project-1.5/Yapa - dev/devel/tests/files/test26.f90 has been modified. Save it ?

The user cannot leave Yapakit with unsaved texts. If he does so, a dialog box allow to save unsave files.

13.4 Content of selection

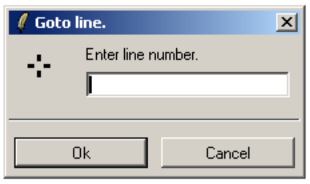


```
irst occurence of mySub26_1 uses mySub26_2,
! the second not.
! If the parsing of the called subroutines is done on the first
! occurence, it is mySub26_2 which is grabbed and mySub26_3 is
unused.
! If the parsing of the called subroutines is done on the second
! occurence, it
```

13.5 Copy and paste

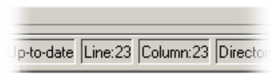
When copying and pasting, one can see the content of the "Clipboard" and the content of the "Selection". The difference between the "Clipboard" and the "Selection" is a feature of the Tk library. When the user selects a part of the text, the selected string is copied into the "Selection". When the user use the "copy" command, the string is moved from the "Selection" into the "Clipboard". This feature can be used when moving text strings between applications.

13.6 Goto line



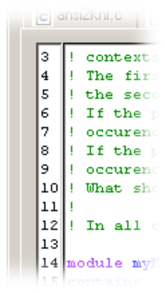
Put the cursor at the specified line number. This is useful when a compiler report a problem for a particular line number, this feature can be useful.

13.7 Line and column number in status bar



The current line and column is displayed in the status bar. It is updated every time the user move the cursor.

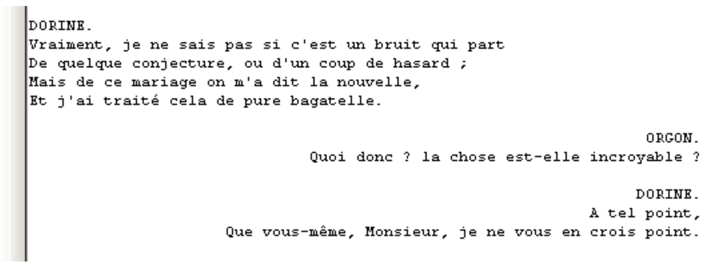
13.8 Line numbers in text widget



```
3  ! context
4  ! The fir
5  ! the sec
6  ! If the
7  ! occuren
8  ! If the
9  ! occuren
10 ! What sh
11 !
12 ! In all
13
14 module my
```

The line numbers can be displayed at the left of the text. This feature is available from the ”Customize > Line numbers” menu.

13.9 Justify lines

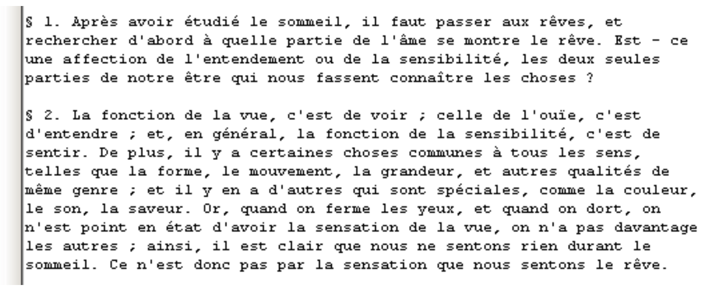


```
DORINE.
Vraiment, je ne sais pas si c'est un bruit qui part
De quelque conjecture, ou d'un coup de hasard ;
Mais de ce mariage on m'a dit la nouvelle,
Et j'ai traité cela de pure bagatelle.

                                                             ORGON.
                               Quoi donc ? la chose est-elle incroyable ?

                                                            DORINE.
                                                       A tel point,
                      Que vous-même, Monsieur, je ne vous en crois point.
```

The commands in the menu ”Edit > Justify... > Justify text to” allow to justify the selected text or lines to plain, left, right or center. When the justification is done, the selected region is then considered as a single string.

The commands in the menu ”Edit > Justify... > Justify lines to” allow to justify all lines of the selected region, where each line is justified separately.

13.10 Fill text



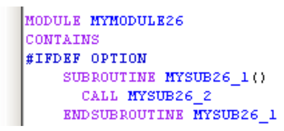
```
§ 1. Après avoir étudié le sommeil, il faut passer aux rêves, et
rechercher d'abord à quelle partie de l'âme se montre le rêve. Est - ce
une affection de l'entendement ou de la sensibilité, les deux seules
parties de notre être qui nous fassent connaître les choses ?

§ 2. La fonction de la vue, c'est de voir ; celle de l'ouïe, c'est
d'entendre ; et, en général, la fonction de la sensibilité, c'est de
sentir. De plus, il y a certaines choses communes à tous les sens,
telles que la forme, le mouvement, la grandeur, et autres qualités de
même genre ; et il y en a d'autres qui sont spéciales, comme la couleur,
le son, la saveur. Or, quand on ferme les yeux, et quand on dort, on
n'est point en état d'avoir la sensation de la vue, on n'a pas davantage
les autres ; ainsi, il est clair que nous ne sentons rien durant le
sommeil. Ce n'est donc pas par la sensation que nous sentons le rêve.
```

Fill (adjust) or unfill text up to the 72-th column. This algorithm takes into account the paragraphs of the selected region so that distinct paragraphs remain distinct.

13.11 Split file into parts

Split a long text file (for example a log file) into several files.

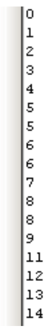
13.12 Upcase / downcase



```
MODULE MYMODULE26
CONTAINS
#IFDEF OPTION
    SUBROUTINE MYSUB26_1()
      CALL MYSUB26_2
    ENDSUBROUTINE MYSUB26_1
```

Upcase/downcase/capitalize region.

13.13 Sort lines



```
0
1
2
3
4
5
5
6
6
7
8
8
9
11
12
13
14
```

Sort lines alphabetically in increasing or decreasing order. Sort lines and keep only unique lines.

The ”Edit > More ... > Delete blank lines” allows to delete all blank lines in the current text. The ”Edit > More ... > Compact text” command allow to remove double blank lines in the current text so that the final text look more ”compact”. ”Edit > More ... > Set double interline” command insert a blank line between two lines.

Chapter 14

Shortcuts, keyboard and mouse bindings

This section presents the shortcuts for Yapakit as well as the interaction with the keyboard and the mouse. The shortcuts and the response of the software to keyboard and mouse events are differents depending on the widget considered :

- the menu bar, in which the user can select commands,
- the text widget, in which the user can edit his files,
- the terminal widget, in which the user can see the results for executed commands, search commands, debug logs.

Of course, the most detailed paragraph in this page is dealing with the editing of files, because Yapakit is mainly an editor.

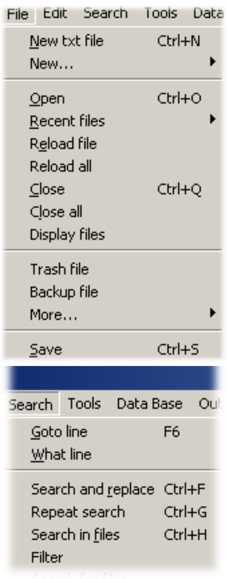
Before going into details, it must be said there is a very limited number of shortcuts in Yapakit. Furthermore, is must be clear that there is no way for the user to define his own shortcuts. There is two reasons for that. The first is that although shortcuts are making the editing process faster,they are occupying a place in my human memory and that space is so precious that I like to use it in a better way. The second reason is that shortcuts are not easy to know because each software has often its own shortcuts. During one day, I use at least a dozen of different softwares and each one has its own shortcuts so that if I want to remember all I have to take drugs (or at least a lot of coffee)...

In Yapakit, the number of shortcuts is the least possible and each shortcut is the most classical possible. If not, the shortcuts are detailed here.

14.1 The menubar

Items in the menubar have one of their letters underline. For example,the ”F” letter in the ”File” menu is underlined. If the user press Alt-f, the menu opens. Then he can select one menu item by using the Up and Down arrow keys. The selection of one command is done thanks to the Enter key. The user can access to sub-menus thanks to Left and Right arrow keys.

In the menu ”File”, the ”New” command is followed by ”Control-N” which means that Control-n is a shortcut for the ”New” commmand.



14.2 Editing

The classical interactions with the keyboard are avalaible : if you press ”a”, you get an ”a”, which I hope will not surprise you. Special characters like ”é” or ”ê” are available if your keyboard allows you to type these kind of letters.

The cut, copy and paste actions are available through the following shorcuts :

- Control-c : copy,
- Control-v : paste,
- Control-x : cut,
- Control-z : undo.

If the text has a selection, it is copied into the Tk’s primary selection. You can paste it with the ”Insert” key of the keyboard.

Moving in the text is allowed thanks to the arrows (Up, Down, Left and Right) and the Scroll Up and Down keys. The mouse wheel is taken into account. Faster scrolling is avalaible thanks to these two shortcuts :

- Control-Home : move the insertion cursor to the beginning of the text and clear any selection in the widget.

- Control-End : move the insertion cursor to the end of the text and clear any selection in the widget.

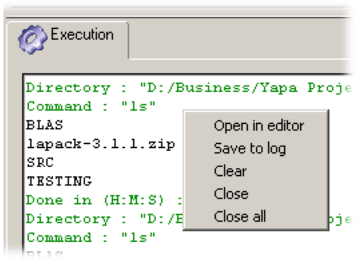
Moreover, the following actions are avalaible :

- ButtonPress-3 (the right button of the mouse) : makes the context popup window appear,
- TAB : indent the current line (context sensitive). The TAB key is bound to the automatic indentation and cannot allow to insert a real TAB. Instead, you can insert an TAB at the current cursor position with Control+i. This is useful in a Makefile, for example.

14.3 Terminals

No editing is allowed directly in a terminal. Instead, the right button of the mouse allow you to acces to a popup menu containing the following items :

- Open in editor,
- Save to log,
- Clear,
- Hide,
- Hide all.

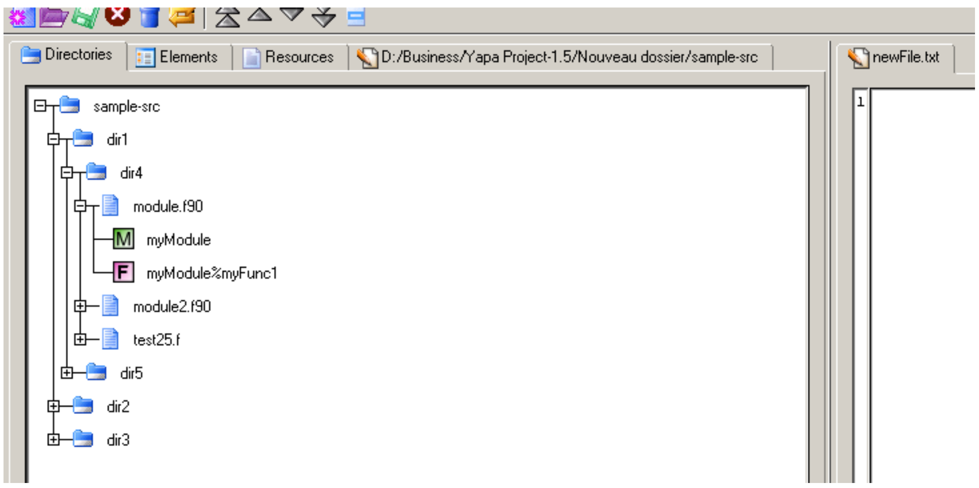


```
Directory : "D:/Business/Yapa Proje
Command : "ls"
BLAS
lapack-3.1.1.zip
SRC
TESTING
Done in (H:M:S) :
Directory : "D:/E
Command : "ls"
```

The text content of the terminal can be opened in the editor and edited (and printed) afterwards like any other text.

14.4 Browsing

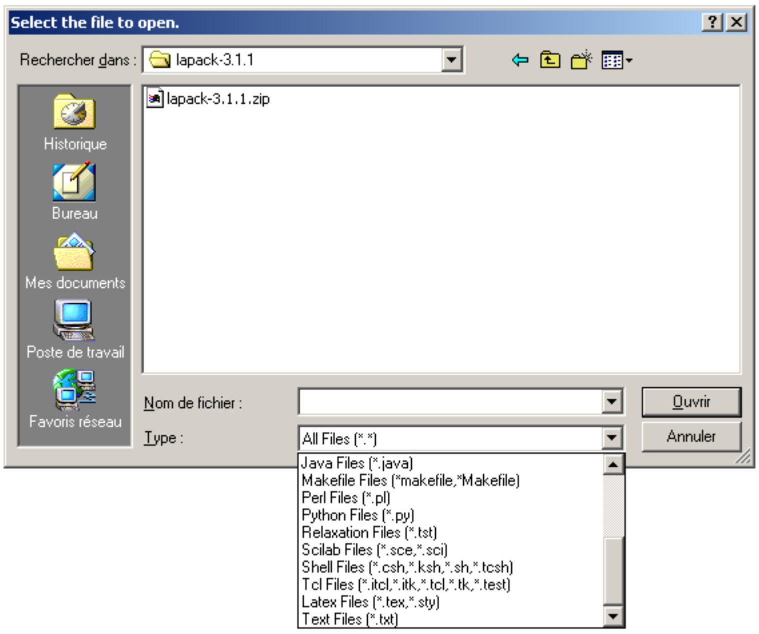
When browsing a database, the tree is responding interactively with actions of the mouse. The left button will open the tree if the user points on the cross and close the tree if the user points on the minus. If the user clicks over a directory (or file), the report corresponding to the directory will be displayed.



14.5 Opening a file

When the user executes the command "Open file", a dialog box appears. The following shortcuts are avalaible.

- TAB : focus the next button in the dialog box
- Esc : close the dialog box
- Enter : open the selected file



14.6 Filling an entry

One can use the classical cut, copy and paste thanks to Control-x, Control-c and Control-v.

Part IV

Fortran Features

Chapter 15

Fortran mode

The fortran is the target of the Yapakit editor, that is why it is worth to describe some details on the fortran mode. The Yapakit editor takes into account the fortran 77 and fortran 90 languages separately.

15.1 File patterns

In Yapakit, the mode is computed from the extension of the file.

The file patterns associated with fortran 90 mode are the following :

- .f90
- .F90
- .f95
- .F95
- .f03
- .F03

The file patterns associated with the fortran 77 mode are the following :

- .f
- .F
- .f77
- .F77
- .for
- .FOR

- .ftn
- .FTN

There is currently no way of customizing these associations between a file extension and an editing mode. If the user has a file with a special fortran file extension which is not recognized by Yapakit as a fortran file (for example ”myfile.myfortranextension”), the user should rename the file and modify the file extension.

15.2 Highlighting

The highlighting is based on the identification in the source code of different types of ”tokens”. These tokens are the following :

- keywords are the basic tokens of the language,
- standard library functions extends the language and provide higher-level features,
- comments,
- strings.

In fortran, there is no standard library, so the distinction between a keyword and a standard library function is somewhat arbitrary. Fortran 77 keywords are the following :

AND ASSIGNMENT CALL CHARACTER COMMON COMPLEX CONTINUE CYCLE DIMENSION DO DOUBLE ELSE ELSEIF END ENDDO ENDIF ENDPROGRAM ENDPROGRAM ENDSUBROUTINE EQ EQUIVALENCE EXIT EXTERNAL FALSE FORMAT FUNCTION GE GOTO GT IF IMPLICIT INCLUDE INTEGER INTRINSIC KIND LE LOGICAL LT NE NONE NOT OR PARAMETER PARAMETER PAUSE PRECISION PRECISION PRINT PROGRAM REAL RECURSIVE RESULT RETURN SAVE SUBROUTINE THEN TO TRUE WHILE BLOCK DATA

Fortran 90 keywords are the fortran 77 keywords plus the following keywords :

CASE CONTAINS ENDINTERFACE ENDMODULE ENDTYPE INTENT INTERFACE MODULE ONLY OPERATOR POINTER PRIVATE PROCEDURE PUBLIC SELECT SEQUENCE TYPE USE ALLOCATABLE OPTIONAL

Fortran 77 standard library functions list is the following:

ABS ACOS AIMAG AINT ALOG ALOG10 AMAX0 AMAX1 AMIN0 AMIN1 AMOD ANINT ASIN ASIND ATAN ATAN2 ATAN2D ATAND BACKSPACE CABS CCOS CHAR CLOG CLOSE CMPLX CONJG COS COSD COSH CPU_TIME CSIN

CSQRT DABS DACOS DASIN DATAN DATAN2 DBLE DCMPLX DCOS DCOSH DDIM DEXP DIM DINT DLOG DLOG10 DMAX1 DMIN1 DMOD DNINT DPROD DREAL DSIGN DSIN DSINH DSQRT DTAN DTANH ENDFILE ERF ERFC EXP FLOAT FLUSH GAMMA GETENV HFIX IABS ICHAR IDIM IDINT IDNINT IEOR IFIX ILEN IMAG INDEX INQUIRE INT INT2 ISIGN LEADZ LEN LGAMMA LGE LGT LLE LLT LOC LOG LOG10 LSHIFT MAX MAX0 MAX1 MIN MIN0 MIN1 MOD NINT NULL NUM_PARTHDS NUM_USRTHDS OPEN QCMPLX QEXT RAND READ REAL REWIND RSHIFT SIGN SIGNAL SIN SIND SINH SIZEOF SNGL SQRT SRAND SYSTEM TAN TAND TANH TRIM WRITE STOP

Fortran 90 standard library functions list is made of the fortran 77 standard library functions list plus the following :

ACHAR ADJUSTL ADJUSTR ALL ALLOCATE ALLOCATED ANY ASSOCIATED BIT_SIZE BTEST CEILING COUNT CSHIFT DATE_AND_TIME DEALLOCATE DIGITS DOT_PRODUCT EOSHIFT EPSILON EXPONENT FLOOR FRACTION HUGE IACHAR IAND IBCLR IBITS IBSET IOR ISHFT ISHFTC KIND LBOUND LEN_TRIM LOGICAL MATMUL MAXEXPONENT MAXLOC MAXVAL MERGE MINEXPONENT MINLOC MINVAL MODULO MVBITS NEAREST NOT NULLIFY NUMBER_OF_PROCESSORS PACK PRECISION PRESENT PROCESSORS_SHAPE PRODUCT RADIX RANDOM_NUMBER RANDOM_SEED RANGE REPEAT RESHAPE RRSPACING SCALE SCAN SELECTED_INT_KIND SELECTED_REAL_KIND SET_EXPONENT SHAPE SIZE SPACING SPREAD SUM SYSTEM_CLOCK TINY TRANSFER TRANSPOSE UBOUND UNPACK VERIFY

Fortran 90 or fortran 77 strings can be made of single or double quotes.

Fortran 90 comment lines are beginning with a "!", but everything after a "!" on a line is considered as a comment.

A fortran 77 line is a valid comment if the first string is one of the following characters : "c", "C" or the star character "*".

15.3 Fortran 77 source forms

There are two formats for fortran 77 language :

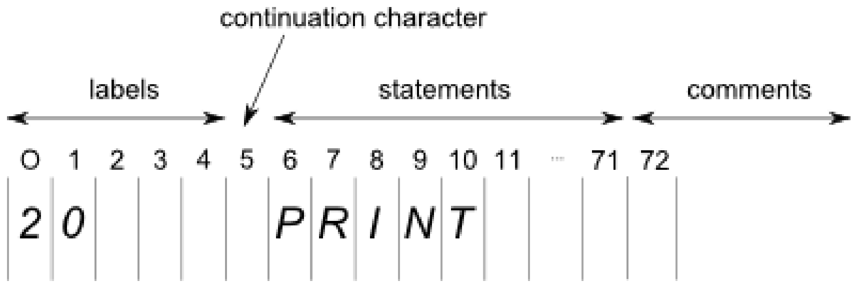
- the fixed format is the standard,
- the TAB-format is an extension of the fortran 77 norm.

Although the TAB-format is no standard fortran, it is accepted by many compilers (gfortran for example) and is taken into account in Yapakit. The current section details the different source code forms which are taken into account in the editor. These details are available in good fortran books, but some insight on what Yapakit considers as a valid fortran statement can be interesting in some situations.

15.3.1 Fortran 77 fixed format

In this form, the line is decomposed into several fields, depending on the column number of the character on the line :

- columns 0 to 4 are containing the labels,
- column 5 is containing the continuation character,
- column 6 to 71 are containing the statements,
- columns 72 to the end are comments.



The continuation character can be any character.

15.3.2 Fortran 77 TAB-format

This format was used at a time where the editors were not able to indent automatically the source code. Notice that even recent editors like Microsoft Visual Studio, Eclipse / Photran does not implement automatic indentation.

The line is considered tab-format one of the following two conditions is satisfied:

- case (A) : there is a tab in columns 0 to 5,
- case (B) : there is a tab in column 0, followed by a digit 1, 2, ... or 9.

If the line is not tab-format, it is expected to be in fixed format.
If the line is in tab-format,

- case (A) : the statement starts from the tab to the end of the line.
- case (B) : the statement, which is a continuation line, starts from the column 2 (column 0 is for the tab, column 1 is for the continuation digit).

Notice that case (A) includes that one label may be followed by a tab as in this example (since tabulations are by definition invisible, I replace it by the <TAB> string) :

```
20<TAB>print *, "toto"
```

15.3.3 Examples

- fixed format :

```
20    print *, 'toto'
```

- tab-format, case (A) :

```
<TAB>print *, 'toto'
```

- tab-format case (B) :

```
<TAB>print *, 'toto',
<TAB>1 "tata"
```

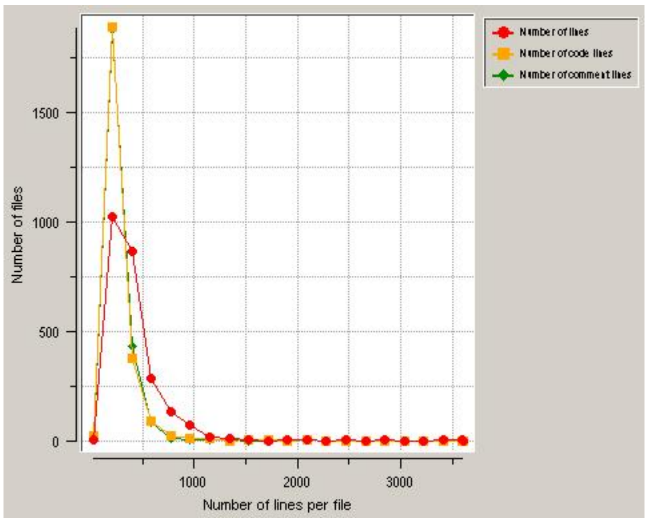
15.4 Fortran 90 element names

For each fortran 90 module, elements in that module are named with ”nameOfModulefunctions, interfaces, derived types). This allow to distinguish between subroutines which have the same name but which are located in different modules.

Chapter 16

Distribution graphs

16.1 Understand lines distribution graph



The distribution is a x-y diagram :

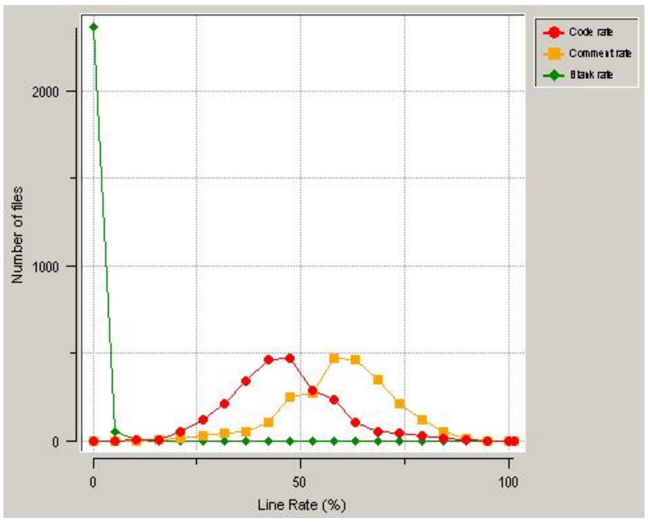
- on the x axis, there is the number of lines,
- on the y axis, there is the number of files which contains the corresponding number of lines.

In fact, three distributions are displayed at the same time:

- the number of lines (comment, code or blank lines),
- the number of comment lines,
- the number of code lines.

The distribution associated with the blank lines is not displayed. For example, considering the number of lines distribution, if there is a point with x=1000 lines and y=50 files, this means that there are 50 files which contains around 1000 lines. The distribution is computed by identifying the minimum and the maximum number of lines (or code lines, or comment lines) in all files. This range is divided in 100 intervals. Then, for each interval, we count the number of files which fall in that interval. This is done by using the "::math::statistics::histogram" computation routine of TclLib. The distribution is then displayed by BLT.

16.2 Understand rates distribution graph



The distribution is a x-y diagram :

- on the x axis, there is the rate,
- on the y axis, there is the number of files which contains the corresponding rate.

In fact, three distributions are displayed at the same time:

- the rate of code lines,
- the rate of comment lines,
- the rate of blank lines.

For example, considering the code rate distribution, if there is a point with x=50are 50 files which contains around 50distribution is computed by dividing the [0intervals.

Then, for each interval, we count the number of files which fall in that interval. This is done by using the "::math::statistics::histogram" computation routine of TclLib. The distribution is then displayed by BLT.

Chapter 17

Convert f77 to F90

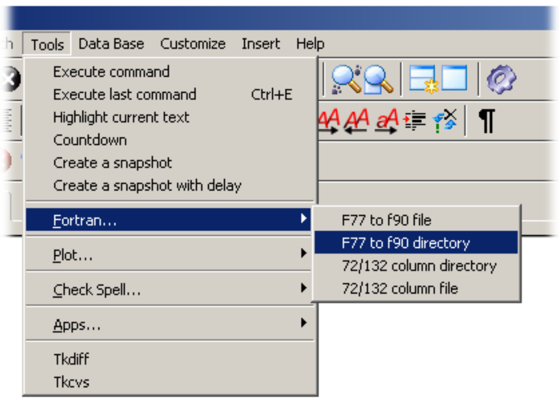
The editor has a convert feature which allows to convert a set of files stored in a directory from the Fortran 77 fixed format source code into the free format Fortran 90 source code. The features of this tool are the following :

- easy-to use and simple graphical Wizard guides the user through the steps,
- convert a whole set of files in a directory and in sub-directories,
- takes into account 2 Fortran 77 formats : fixed or tab format,
- convert automatically from Fortran 77 comments to Fortran 90 comments,
- does not modify pre-processing directives (like ”#ifdef...” for example),
- takes into account for continuation lines and use ”&” as the continuation Fortran 90 character,
- does not modify Fortran 77 labels so that GOTO statements can be used in the new Fortran 90 source code.

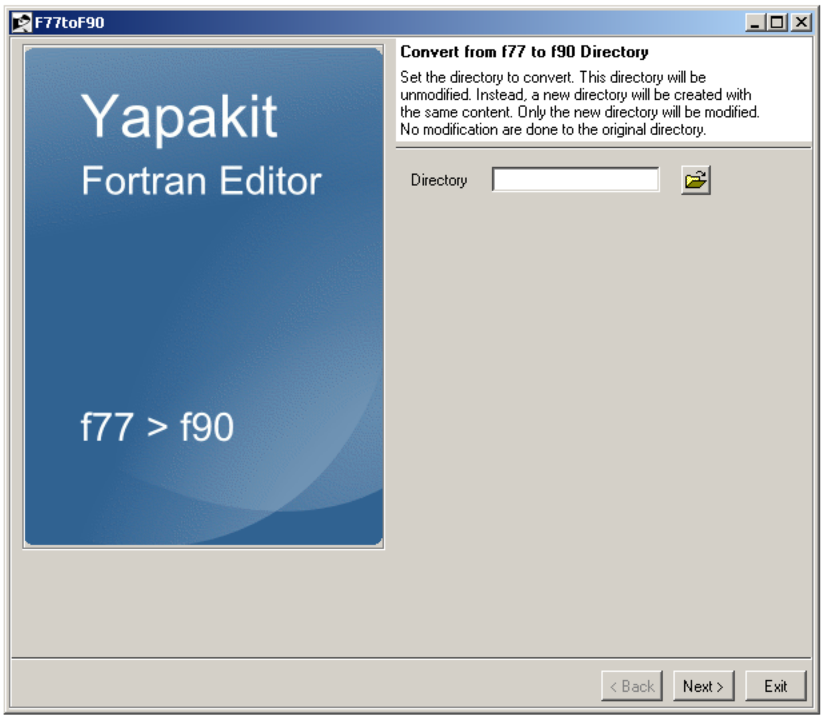
The current page contains a description of the Wizard which guides the user through the steps. It contains also a demo of the tool in action. The provided zip file contains a sample set of Fortran 77 files on which the user can test the conversion tool.

17.1 Using the converter

This feature can be found in the ”Tools” menu, under the ”Fortran sub-menu.



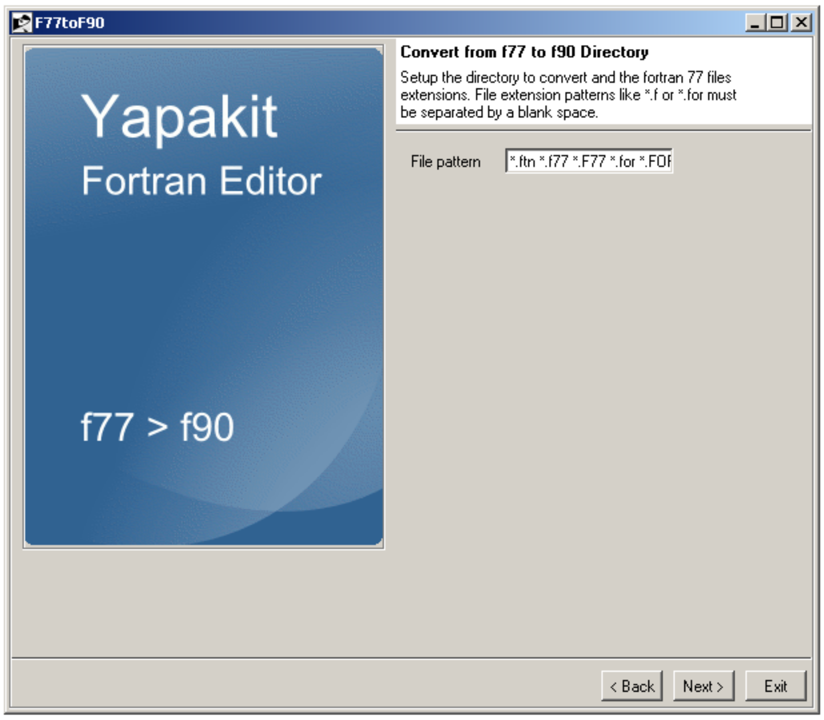
Once selected, the f77tof90 dialog appears. This ”Wizard” will guide you through the steps. The first step consists in selecting the directory to convert. This original directory will not be modified. Instead, a new directory will be created by making a copy of the original directory. The conversion will be performed onto the new directory. Once the directory has been set, click on ”Next”.



Convert from f77 to f90 Directory

Set the directory to convert. This directory will be unmodified. Instead, a new directory will be created with the same content. Only the new directory will be modified. No modification are done to the original directory.

The second step consists in setting the Fortran 77 file extensions. The files matching that list of patterns will be converted. Once the patterns are set, click on ”Next.”

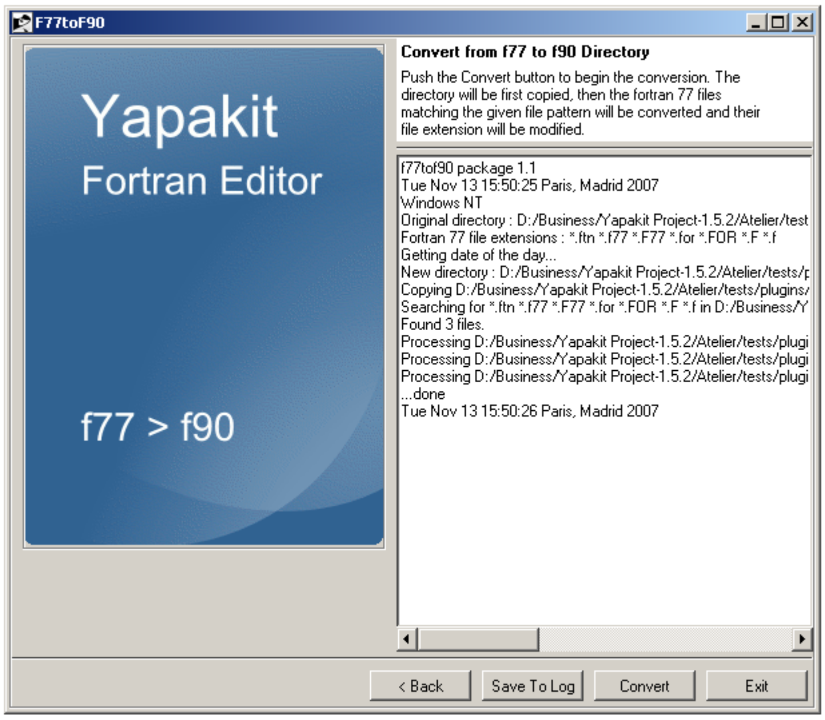


Convert from f77 to f90 Directory

Setup the directory to convert and the fortran 77 files extensions. File extension patterns like *.f or *.for must be separated by a blank space.

File pattern: *.ftn *.f77 *.F77 *.for *.FOF

The last page of the Wizard is the logger. To begin the conversion, click on the ”Convert” button. The text will be filled with the messages, which indicate the progression of the tool. Once the conversion is performed, the user can click on the ”Save To Log” button to save the text into a logfile. Once finished, click on ”Exit”.



Convert from f77 to f90 Directory

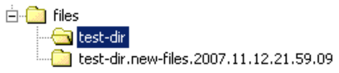
Push the Convert button to begin the conversion. The directory will be first copied, then the fortran 77 files matching the given file pattern will be converted and their file extension will be modified.

```
f77tof90 package 1.1
Tue Nov 13 15:50:25 Paris, Madrid 2007
Windows NT
Original directory : D:/Business/Yapakit Project-1.5.2/Atelier/test
Fortran 77 file extensions : *.ftn *.f77 *.F77 *.for *.FOR *.F *.f
Getting date of the day...
New directory : D:/Business/Yapakit Project-1.5.2/Atelier/tests/p
Copying D:/Business/Yapakit Project-1.5.2/Atelier/tests/plugins/
Searching for *.ftn *.f77 *.F77 *.for *.FOR *.F *.f in D:/Business/Y
Found 3 files.
Processing D:/Business/Yapakit Project-1.5.2/Atelier/tests/plugi
Processing D:/Business/Yapakit Project-1.5.2/Atelier/tests/plugi
Processing D:/Business/Yapakit Project-1.5.2/Atelier/tests/plugi
...done
Tue Nov 13 15:50:26 Paris, Madrid 2007
```

17.2 Sample use

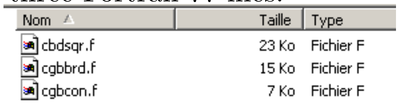
The following picture is a capture of the test directory, for which a zip archive can be found at : test-dir.zip

You can use it to test the capabilities of the converter. The tested files are extracted from Lapack. As you can see in the picture, the new directory, containing the new Fortran 90 files, is named ”test-dir.new-files.(date of conversion)”.



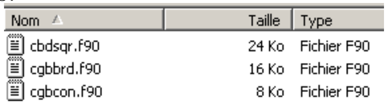
```
files
  test-dir
  test-dir.new-files.2007.11.12.21.59.09
```

The following picture is the content of the original ”test-dir” directory, which contains the three Fortran 77 files.



Nom	Taille	Type
cbdsqr.f	23 Ko	Fichier F
cgbbrd.f	15 Ko	Fichier F
cgbcon.f	7 Ko	Fichier F

The following picture is the content of the new directory, with the new Fortran 90 files.



Nom	Taille	Type
cbdsqr.f90	24 Ko	Fichier F90
cgbbrd.f90	16 Ko	Fichier F90
cgbcon.f90	8 Ko	Fichier F90

Part V

Plugins

Chapter 18

Plugins management

This section explains how the plugins are managed in Yapakit. In the first section, one details the general principles of this management, especially the way of enabling / disabling the plugins. The next two sections describe the way that plugins are stored in the Yapakit directory structure and the content ("anatomy") of one sample plugin. One section detail how the plugins are loaded into Yapakit. The last two sections focus on how to remove or add a plugin into Yapakit.

18.1 How to get plugins

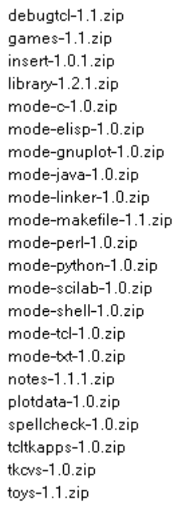
The currently available plugins are available from the "Download" page on the Yapakit web site :

```
\yapakitwebpage/Download.html
```

For example, the current plugins zip file is the following :

```
\yapakitwebpage/zip-plugins2007.11.24.15.47.13.zip
```

In the zip file is stored a directory containing all the plugins available, as zip file again. Installing a plugin simply consists in unzipping the corresponding zip file and copy it into the "plugins" sub-directory of the Yapakit installation directory.



```
debugtcl-1.1.zip
games-1.1.zip
insert-1.0.1.zip
library-1.2.1.zip
mode-c-1.0.zip
mode-elisp-1.0.zip
mode-gnuplot-1.0.zip
mode-java-1.0.zip
mode-linker-1.0.zip
mode-makefile-1.1.zip
mode-perl-1.0.zip
mode-python-1.0.zip
mode-scilab-1.0.zip
mode-shell-1.0.zip
mode-tcl-1.0.zip
mode-txt-1.0.zip
notes-1.1.1.zip
plotdata-1.0.zip
spellcheck-1.0.zip
tcltkapps-1.0.zip
tkcvs-1.0.zip
toys-1.1.zip
```

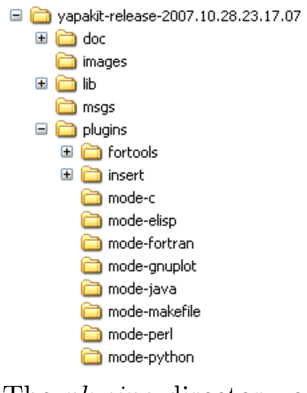
18.2 General Principles

The Yapakit editor integrates a plugin system which allows to configure easily the editor with additional features which does not need a modification of the core. These plugins can be enabled or disabled by a simple creation of destruction of the directory containing that plugins. Some plugins are already in the release of the editor but additional plugins will be available as separate downloads.

There is no current way for users to develop their own plugins, since the API of the editor is not public. The goal of the current page is just to allow you to configure the editor features, by understanding how plugins are managed so that you can add or remove a plugin as you wish.

18.3 Plugins Location

The following picture shows the content of the release directory.

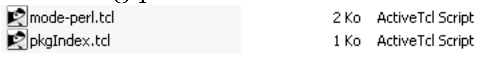


```
yapakit-release-2007.10.28.23.17.07
    doc
    images
    lib
    msgs
    plugins
        fortools
        insert
        mode-c
        mode-elisp
        mode-fortran
        mode-gnuplot
        mode-java
        mode-makefile
        mode-perl
        mode-python
```

The *plugins* directory contains Tcl packages which are stored in directories. There is exactly one package by sub-directory.

18.4 Anatomy of a Plugin

The following picture shows the content of the ”mode-perl” plugin.



```
mode-perl.tcl        2 Ko   ActiveTcl Script
pkgIndex.tcl         1 Ko   ActiveTcl Script
```

The file ”pkgIndex.tcl” is the package index for the directory. The file ”mode-perl.tcl” contains the script of the plugin.

18.5 How Plugins are loaded

At the startup of the editor, the list of all sub-directories of the emphplugins directory is computed. That list of directories is added to the Tcl global ”::auto_path” variable, so that Tcl knows that these directories may contain packages. The name of the directory may be different from the name of the package. This allows that the directory name may include a version number to makes the manual management of the plugins easy. The editor loads the plugin by using the ”package require packagename” Tcl command. Each plugin directory may contain one or several Tcl files, and may contain sub-directories, in a way that is internal to the plugin. The only requirement is that the ”pkgIndex.tcl” located at the root of the plugin directory defines the list of Tcl scripts to source in order to load a package. The name of the package associated with the plugin is computed from the content of the ”pkgIndex.tcl” file.

18.6 How to Remove a Plugin

Removing the plugin ”pluginName” simply consists in deleting the directory ”plugins/pluginName”.

18.7 How to Add a Plugin

Adding a new plugin ”pluginName” simply consists in moving the directory under the ”plugins” directory.

Chapter 19

Available Plugins

The current page is a full list of available plugins for Yapakit. Plugins add one or several features to Yapakit without any change to the core of Yapakit. Each plugin is made of one or two parts :

- a Yapakit Plugin Script,
- and, in some cases, external softwares.

The licence associated with Yapakit Plugin Script is the same as the Yapakit Licence. The external softwares remain the property of their respective authors.

The details of how to get and install the plugins is explained in the ”Plugins Management” section of this document.

The full list of available plugins is the following :

- debugtcl-1.1
- games-1.1
- insert-1.0
- library-1.2.1
- mode-c-1.0
- mode-elisp-1.0
- mode-gnuplot-1.0
- mode-java-1.0
- mode-linker-1.0
- mode-makefile-1.1
- mode-perl-1.0

- mode-python-1.0
- mode-scilab-1.0
- mode-shell-1.0
- mode-tcl-1.0
- mode-txt-1.0
- notes-1.1.1
- plotdata-1.0
- spellcheck-1.0
- tcltkapps-1.0
- tkcvs-1.0
- toys-1.1

19.1 debugtcl-1.1

This plugin allows to debug a Tcl project with various Tcl/Tk tools. These tools remain the property of their respective authors. This plugin offers facilities to :

- check Tcl syntax of one script with Frink and TDK checker,
- check Tcl syntax of one directory with Frink and TDK checker,
- debug a Tcl script with Tkinspect or RamDebugger,
- open the Tcl Console,
- execute Tcl unit tests.

It includes the following tools :

- Frink 2.2.2 patch level 4 in source and binary form for Linux, Sun Solaris Sparc and Linux,
- RamDebugger-5.3b
- Tclunit by Bob Techentin, October 24, 2005
- tkinspect-5.1.6p10

This plugin adds the "Debug Tcl..." entry in the Tools menu of Yapakit.

19.2 games-1.1

This plugin allows to play with several Tcl/Tk games. These softwares remain the property of their respective authors. This plugin adds the ”Games...” entry in the Tools menu of Yapakit.

19.3 insert-1.0

This plugin offers several facilities to insert texts, templates and special characters into Yapakit. This plugin offers facilities to :

- insert the current data, your Name, your E-Mail into the current text,
- insert the content of another file into the current text,
- Tcl/Tk templates into the current text,
- Fortran 90 templates into the current text,
- Makefile templates into the current text,
- Yapakit bug report and feature request templates,
- special characters such as tabulation or carriage return.

The Name, E-Mail, Phone number and Project Name can be customized in the general Customize feature of Yapakit.

19.4 library-1.2.1

This plugin allows to manage a simple database of CD or DVD. The goal is to provide a simple (some will say ”too simple”) way of finding a particular file stored in a collection of backup CD or DVD, where each CD/DVD is identified with a particular label (for example ”CD#1”, ”CD#2,” etc...). The user can store that database into a .csv file and add the table of content of the CD or DVD into the current database. Do not worry : it is not the whole content of the CD which is copied into the .cvs file ! Each file found in the CD or DVD is analysed and its file name is registered in the database. Other additionnal informations such as file size and type of file are also computed. Each file is associated with a category, depending on its file extension. These are the currently available categories :

- Software : .EXE,
- Sound : .CDA, .WAV, .WMA, .MP2, .MP3,
- Text : .TXT,

- Picture : .JPG, , .GIF, .PSD, .JPEG,
- Film : .AVI, .MPEG, .WMV, .MOV, .MPG,
- File : anything else.,

The tool allows to filter all files matching a given category. A typical collection of around 100 CD is stored in a .csv file which size is less than 500 kB.

19.5 mode-c-1.0

This plugin gives the highlighting feature of C/C++ source code to Yapakit. The following default association is done between the file extension and the file mode :

- C : *.c, *.h
- CPP : *.cpp

19.6 mode-elisp-1.0

This plugin gives the highlighting feature of Emacs-Lisp source code to Yapakit. The following default association is done between the file extension and the file mode :

- elisp : *.el

19.7 mode-gnuplot-1.0

This plugin gives the highlighting feature of Gnuplot scripts to Yapakit. The following default association is done between the file extension and the file mode :

- Gnuplot : *.gp

19.8 mode-java-1.0

This plugin gives the highlighting feature of Java source code to Yapakit. The following default association is done between the file extension and the file mode :

- Java : *.java

19.9 mode-linker-1.0

This plugin allows to parse messages generated from linkers like ld. Messages like ”undefined reference to” are captured by Yapakit in such a way that the user can click onto messages and the editor opens the corresponding file and sets the cursor at the right position.

19.10 mode-makefile-1.1

This plugin gives the highlighting feature of Makefile files to Yapakit. The following default association is done between the file extension and the file mode :

- Makefile : *makefile*, *Makefile*

Tabulations plays a special role in Makefiles. The user should not use the keyboard key ”Tab” to insert a tabulation into Makefiles, because that key is bound to the indent feature of the editor. Instead, one can use the keyboard sequence ”Control+i” to insert a real tabulation. In order to see further details in the Makefile, one can use the ”display invisible characters” feature so that tabulations are clearly displayed.

19.11 mode-perl-1.0

This plugin gives the highlighting feature of Perl scripts to Yapakit. The following default association is done between the file extension and the file mode :

- Perl : *.pl

19.12 mode-python-1.0

This plugin gives the highlighting feature of Perl scripts to Yapakit. The following default association is done between the file extension and the file mode :

- Python : *.py”

19.13 mode-scilab-1.0

This plugin gives the highlighting feature of Scilab scripts to the editor. The following default association is done between the file extension and the file mode :

- Scilab : *.sce, *.sci

19.14 mode-shell-1.0

This plugin gives the highlighting feature to Shell scripts to the editor. The following default association is done between the file extension and the file mode :

- Shell : *.csh, *.ksh, *.sh, *.tcsh, .*rc*

19.15 mode-tcl-1.0

This plugin gives the highlighting feature to Tcl/Tk scripts to the editor. The following default association is done between the file extension and the file mode :

- Tcl : *.itcl, *.itk, *.tcl, *.tk, *.test

19.16 mode-txt-1.0

This plugin gives the highlighting feature to text files to the editor. The following default association is done between the file extension and the file mode :

- Text : *.txt

19.17 notes-1.1.1

This plugin gives a simple way of storing everyday text notes. The goal is to allow the user to lighten their memory and to give a fast and easy way of storing these thoughts. It is some kind of Blog, but with no formatting other than text. The hypothesis is that one note by day is enough to store notes. This plugin adds a ”Notes” menu into Yapakit. Notes are regular text files, which file name contains the date of the day where the notes are written. The user can define a directory in which all notes will be stored (Notes > Customize). At startup, all the entries in the Notes menu are disabled, except the Customize entry. This force the user to define the Notes directory, which is a regular directory of your disk. In the end, this directory may contain 365 files if you write one note every day. If this is the first note of the day, the ”Notes > Write note” opens a new note. If you allready wrote a note previously during the day, the ”Write note” normal behaviour is to open the note of the day. The ”Note > Open recent note” opens the more recent note (in fact, the file which has the greater modification time). The ”Search in notes” features allows to make simple searches into the notes, for example to remember what you did that particular day. You can insert a template note with ”Insert note template”. The ”Display notes” features opens a pane in which the user can display all notes files or display only notes on a particular day. The ”Properties” menu entry opens a dialog box and display general informations about the current notes : directory in which notes are stored, number of notes, file size and more.

19.18 plotdata-1.0

This plugin is a naive attempt at displaying a data file with BLT widgets. This tools adds the entry ”Plot...” in the ”Tools” menu of the editor. It allows to directly plot a two-columns data file with either a X-Y graph or a barchart. The ”sample.txt” file, located inside the plugin directory ”plugins/plotdata-1.0” is a sample text file which contains two columns separated with a blank space. The first line is ”1.0 2.0”, the second line is ”2.0 4.0”, etc... If the user executes the ”Tools > Plot... > Plot Graph”, one dialog box appears, which asks for a data file. If one choose the ”sample.txt” file, a X-Y graph is displayed.

19.19 spellcheck-1.0

This plugin allows to check the spelling of text files. It adds the ”Check Spell...” entry in the ”Tools” menu of the editor.

The check is done on the base of one dictionnary, which is a simple list of words. This dictionnary must be stored in a regular text file. Two dictionnaries are provided with the current package :

- English dictionnary : dict/dict-english.txt
- French dictionnary : dict/dict-french.txt

The check algorithm is based on several simple algorithms, inspired by the methods used in ispell. The high-tech methods which can be found in aspell are not provided here.

The ”Check Spell” command allows the check the spelling of the currently opened file. This is done whatever the type of file opened (either text file or not). The words which are not found in the dictionnary are colored. The user can then right-click onto the colored word. A list of suggested words for replacement is then computed, which may take some time, depending on the length of the word. The suggestions are then displayed in the popup window. If the user select one replacement word, the original word is replaced by the new one. To finish the check and remove the spell colors, execute the ”Terminate Checking” command.

The user can select a different dictionnary with the ”Change dict.” command. The ”Properties dict.” displays a dialog which shows the properties of the current dictionnary : file size and number of words. The ”Display dict.” feature allows to get a manual look into the dictionnary. It displays a ”Words” view where the user can display words matching a given word pattern. The case of the word can be taken into account or not. For example, typing ”ach” in the filter entry displays the list of all words matching ”ach*”.

19.20 tcltkapps-1.0

This plugin allows to use several external Tcl/Tk software. While this plugin remain the property of the author of Yapakit, the Tcl/Tk external software are remaining the property of their respective authors. The goad is to provided simple applications to enrich the toolbox for everyday work. The audience is the scientific community (using Fortran for example) which needs to perform computations, draw functions or convert between physical units. This is the full list of software distributed along with the plugin :

- Calc : a simple but efficient calculator
- Tkparam : to display parametrized functions x(u), y(u)
- Convert : to convert between physical international units

- Dirsize : to display the size of directories
- Stopwatch : to display time with start, stop, pause
- Taskspace : to organize your tasks in the Urgency / Importance space
- Zorro : simple to-do list manager
- Tkoutline : to manage tree-based notes including hyperlinks and text formatting

19.21 tkcvs-1.0

This plugin allows to use TkCvs from Yapakit. TkCvs remains the property of their authors.

- TkCvs : manage files from a CVS repository to work in a team,
- Tkdiff : compare two files and see differences.

19.22 toys-1.1

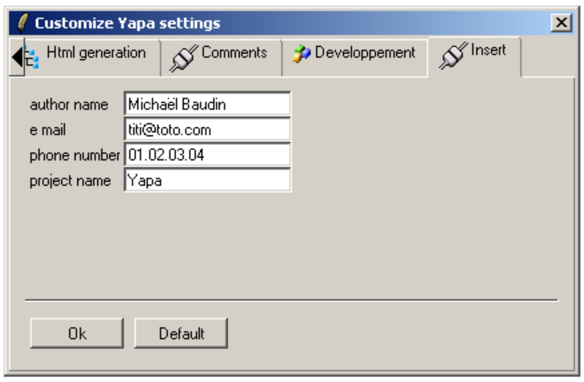
This plugin allows to use several external Tcl/Tk toys.

- Tkfont : displays the available fonts on the system (Unix only)
- Tkgamma : change the gamma of the screen (Unix only)
- Xcolors : display X colors (Unix only)
- Tkwhiteboard : an electronic whiteboard
- Tkcolors : display all Tk colors

Chapter 20

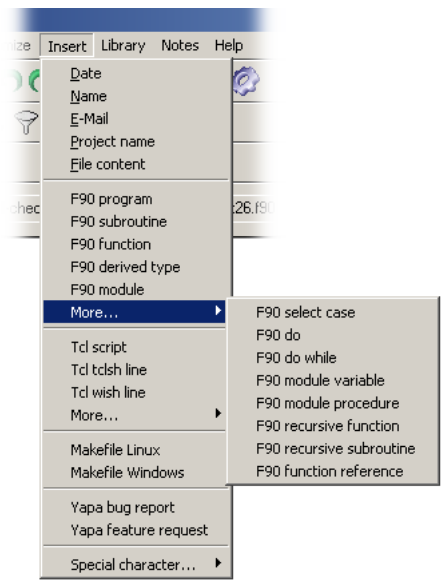
Insert Plugin

In this section, details on the ”Insert” plugin are given. This plugin is, like other plugins, optional in the sense that removing it does not alter the way that the Yapakit core features behave.



author name: Michaël Baudin
e mail: titi@toto.com
phone number: 01.02.03.04
project name: Yapa

The user can insert the date, e-mail, project name, author name. These options can be customized in the ”Customize” dialog.



Several templates for fortran 77, fortran 90, Tcl and Makefile can be inserted into the current file.